TECHNISCHE
UNIVERSITÄT
DARMSTADT

Virtuell Kontextbasierte Dienste

Definition, Beschreibung und Verwendung von Kontext in Virtuellen Welten

Vom Fachbereich
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte

Dissertationsschrift

von

Dipl.-Inf. Sonja Bergsträßer
geboren am 27.04.1977 in Heppenheim an der Bergstraße

Erstreferent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Carsten Griwodz
Korreferent: Prof. Dr.-Ing. Wolfgang Effelsberg

Tag der Einreichung: 23.04.2010
Tag der Disputation: 25.05.2010

Darmstadt, 2010
Hochschulkennziffer D17

Virtuell Kontextbasierte Dienste

Definition, Beschreibung und Verwendung von Kontext in Virtuellen Welten

Zur Erlangung des Grades eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation von Dipl.-Inf. Sonja Bergsträßer
geboren am 27.04.1977 in Heppenheim an der Bergstraße
2010 – Darmstadt – D17

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Elektrotechnik
und Informationstechnik

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Virtuell Kontextbasierte Dienste
Definition, Beschreibung und Verwendung von Kontext in Virtuellen Welten
genehmigte Dissertation von Dipl.-Inf. Sonja Bergsträßer geboren am 27.04.1977 in Heppenheim an der Bergstraße

Tag der Einreichung: 23.04.2010
Tag der Disputation: 25.05.2010

Erstreferent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Carsten Griwodz
Korreferent: Prof. Dr.-Ing. Wolfgang Effelsberg

Technische Universität Darmstadt
Fachbereich Elektrotechnik und Informationstechnik

Fachgebiet Multimedia Kommunikation (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Abstract

Virtuelle Welten haben immer mehr Teilnehmer und gewinnen auch aus wirtschaftlicher Perspektive zunehmend an Bedeutung. Die heute populärsten und in der Literatur am meisten betrachteten virtuellen Welten sind die Spielewelten von Multiplayer Online Games (MOGs). Sie gelten in der Forschung als Vorreiter gemeinsamer Onlinewelten und gehören zu den populärsten und kollaborativsten Applikationen [BB04]. Millionen Spieler aus der ganzen Welt treffen sich in den virtuellen Welten der Multiplayer Online Games und interagieren miteinander. Sie schließen sich in kleinen Teams oder großen Gruppen zusammen und bauen soziale Netzwerke auf. Multiplayer Online Games sind Kern einer komplexen, vernetzten, technischen und sozialen Struktur bei der die Interaktion ein grundlegender Aspekt ist. Dabei geht es um Zusammenarbeit, Gemeinschaft, Austausch und Lernen [NH06]. Informelles Lernen spielt dabei eine zentrale Rolle. Gemischt kollaborative Räume, wie es Massively Multiplayer Online Games (MMOGs) heute schon sind, können auch in anderen Bereichen wie dem interdisziplinären wissenschaftlichen Arbeiten eine gemeinsame Plattform bieten.

In Multiplayer Online Games ist die Interaktion der Spieler von zentraler Bedeutung. Es entstehen Kooperationen zwischen Spielern und es bilden sich Spielergruppen. Die Spieler und vor allem die Spielergruppen interagieren dann nicht nur innerhalb der virtuellen Spielwelt, sondern auch außerhalb des Spiels, in der Community, die sich um das Spiel herum bildet.

Die vorliegende Arbeit basiert auf der Beobachtung, dass verschiedene Funktionen und Informationen neben dem eigentlichen Spiel ergänzend genutzt werden. Dies geschieht, da entsprechende Funktionen und Informationen im Spiel direkt nicht zur Verfügung stehen und die Spieler somit zusätzliche Dienste benötigen. Es bildet sich um das Spiel herum ein Umfeld aus verschiedenen Applikationen und Artefakten, das die Community des Spiels aktiv mitgestaltet. Beispielsweise findet ein Teil der Organisation von Spielergruppen oftmals außerhalb des Spiels statt, da in den Spielen zumeist keine ausreichende Unterstützung hierfür angeboten wird.

Die Dienste im Spielumfeld werden von verschiedenen eigenständigen Internet-Applikationen bereitgestellt. Um diese externen Dienste nutzen zu können, muss der Spieler zwischen dem Spiel und der entsprechenden Applikation wechseln, wodurch das Spielerlebnis gestört wird. Eine umfassende Unterstützung von Spielern innerhalb von virtuellen Welten trägt zu einer Verbesserung des Spielerlebnisses bei. Deshalb verfolgt die vorliegende Dissertation den Ansatz, eine umfassende Unterstützung innerhalb von virtuellen Welt zu ermöglichen. Diese Unterstützung wird durch die Integration externer Dienste in die virtuelle Welt ermöglicht.

Hierzu wurde im Rahmen der Arbeit eine umfassende Analyse virtueller Welten durchgeführt. Basierend auf den Ergebnissen der Analyse wird das Konzept des Kontextes wie es im Gebiet der kontextbewussten Dienste verwendet wird, auf virtuelle Welten übertragen und das Konzept des virtuellen Kontextes wird eingeführt und definiert. Weitere Ergebnisse der durchgeführten Analyse sind die Entwicklung eines generischen Interaktionsmodells für virtuelle Welten und die Identifizierung der virtuellen Parameter als virtuelle Kontextinformationen. Um virtuellen Kontext technisch erfassbar und nutzbar zu machen, wird eine Beschreibungssprache für virtuellen Kontext entwickelt. Anhand der gewonnen Erkenntnisse wird das Konzept der virtuell kontextbasierten Dienste eingeführt und definiert. Virtuell kontextbasierte Dienste ermöglichen eine situationsabhängige Benutzerunterstützung in virtuellen Welten.

Die Anbindung und Integration der externen Dienste wird mittels der Realisierung eines Informationsaustauschs zwischen virtuellen Welten und anderen Internet-Applikationen ermöglicht. Dieser Informationsaustausch basiert auf dem Konzept der virtuell kontextbasierten Dienste und ermöglicht die kontrollierte Weitergabe von Informationen aus der virtuellen Welt an Applikationen im Spielumfeld. Der realisierte Informationsaustausch umfasst die Akquise von virtuellen Kontextinformationen, deren

Verteilung an die verwendeten externen Dienste, sowie die Integration der resultierenden Dienstdaten in die virtuelle Welt.

Es wird eine Middleware-Architektur für die Bereitstellung kontextbasierter Dienste und die Realisierung des Informationsaustauschs konzipiert, umgesetzt und evaluiert. Die Middleware beinhaltet Komponenten zur Erfassung von Kontextinformation in virtuellen Umgebungen, zur Kontextverarbeitung und Verteilung, sowie zum Angebot von Diensten basierend auf der virtuellen Situation des Benutzers und zur Bereitstellung externer Dienste in der virtuellen Welt.

Die im Rahmen der Arbeit durchgeführte Evaluation der umgesetzten Middleware zeigt die Validität des in der Arbeit entwickelten Konzeptes der virtuell kontextbasierten Dienste und die Umsetzbarkeit einer Architektur zur Realisierung des Informationsaustauschs auf Grundlage von kontextbasierten Diensten. Die vorliegende Arbeit schafft also durch die Definition von virtuellem Kontext, dessen Beschreibung sowie dessen Verwendung, basierend auf dem Konzept der virtuell kontextbasierten Dienste, die Grundlage für eine situationsbezogene Benutzerunterstützung innerhalb von virtuellen Welten und für die Integration externer Dienste in virtuelle Welten.

Inhaltsverzeichnis

1 Einleitung

Die heute populärsten und in der Literatur am meisten betrachteten virtuellen Welten sind die Spielewelten von Multiplayer Online Games. Diese bieten durch ihre vielfältigen Aspekte viele interessante Forschungsfragestellungen in unterschiedlichen Forschungsbereichen. Multiplayer Online Games gelten in der Forschung als Vorreiter gemeinsamer Onlinewelten und werden in verschiedenen Bereichen betrachtet, zum Beispiel für die Untersuchung der Netzwerkkommunikation in verteilten Systemen oder um Schlüsse für die Entwicklung und das Design erfolgreicher sozialer Onlinewelten zu ziehen. Für diese Arbeit ist vor allem eine Verknüpfung von virtuellen Spielwelten mit den Aktivitäten im Spielumfeld von Interesse. Betrachtet werden Aspekte, wie die Integration externer Dienste in Multiplayer Online Games, die Nutzung von spielinternen Informationen sowie die Bereitstellung von erweiterten Diensten und Schnittstellen.

1.1 Motivation

Virtuelle Welten, wie sie beispielsweise in Multiplayer Online Games (MOGs) zu finden sind, werden immer beliebter. Sie haben immer mehr Teilnehmer und gewinnen auch aus wirtschaftlicher Perspektive zunehmend an Bedeutung. Millionen Spieler aus der ganzen Welt treffen sich in den virtuellen Welten von Multiplayer Online Games, veranstalten Wettkämpfe, schließen sich in kleinen Teams oder großen Gruppen zusammen und bauen soziale Netzwerke auf.

Dabei sind Multiplayer Online Games "*not just games*" [Yee06], sie gehören zu den populärsten und kollaborativsten Applikationen [BB04]. Multiplayer Online Games sind Kern einer komplexen vernetzten technischen und sozialen Struktur, bei der die Interaktion ein grundlegender Aspekt ist und es geht um Zusammenarbeit, Gemeinschaft, Austausch und Lernen [NH06]. Dabei spielt informelles Lernen eine zentrale Rolle. Gemischt kollaborative Räume, wie es Massively Multiplayer Online Games (MMOGs) heute schon sind, können auch in anderen Bereichen wie dem interdisziplinären wissenschaftlichen Arbeiten eine gemeinsame Plattform bieten.

Multiplayer Online Games wurden für das gemeinsame Spielen konzipiert. Es teilen sich bis zu mehrere Tausend Spieler eine Spielwelt, wobei die virtuellen Repräsentation der Spieler (virtuelle Charaktere) jederzeit miteinander interagieren können. Virtuelle Welten sind komplexe Systeme. Sie bieten dem Teilnehmer vielfältige Interaktionsmöglichkeiten und unterschiedliche Aufgaben an. Der Teilnehmer kann die Ziele, die er erreichen möchte, entsprechend seiner Vorlieben selbst definieren. In Multiplayer Online Games ist die Interaktion der Spieler von zentraler Bedeutung. Es entstehen Kooperationen zwischen Spielern und es bilden sich Spielergruppen. Die Spieler und vor allem die Spielergruppen interagieren dann nicht nur innerhalb der virtuellen Spielwelt, sondern auch außerhalb des Spiels in der Community, die sich um das Spiel herum bildet. Gerade auch die Organisation der Spielergruppen findet teilweise außerhalb in der Umgebung des Spiels statt, da in den Spielen zumeist keine ausreichende Unterstützung hierfür angeboten wird.

Multiplayer Online Games benötigen eine aktive Beteiligung der Spieler. Ihr Erfolg ist abhängig von der Lebendigkeit der Gaming-Community. Besonders in den persistenten Welten der MMOGs sind die Teilnehmer nicht einfach nur Spieler, sie werden ein Bestandteil der virtuellen Welt. Das Vorhandensein anderer, die gemeinsamen Aktivitäten und die entstehenden Communities machen ein MOG erst aus [DYNM06]. Der Austausch über das Spiel, seinen Zustand und seine Entwicklung, spezielle Events und das Organisieren von Aktivitäten in und durch die Communities sind wichtige Aspekte der Spielerfahrung. Auch aus diesem Grund schließen sich die Spieler in festen Gruppen zusammen, knüpfen soziale Netzwerke und tauschen Informationen aus. Die Spieler werden Teil der Gaming-Community des Spiels

und gestalten dadurch das Umfeld des Spiels mit. Sie erstellen "*Community Knowledge*" in kollektiver Wissensproduktion und erweitern dadurch die Grenzen des Spiels [Tay03]. Das Umfeld des Spiels enthält Applikationen, die den Spielern zusätzliche Informationen und Funktionen anbieten, die sie vor, nach oder während des Spielens nutzen. Zu diesen von den Applikationen bereitgestellten externen Diensten gehören Informationsdienste, Kommunikationsdienste und Kollaborationsdienste.

Obwohl sich ein Teil der Spielaktivität damit aus dem eigentlichen Spiel heraus verlagert, ist das Eintauchen eines Spielers in das Spiel (Immersion) entscheidend für sein Spielerlebnis. Diese Immersion und damit das Spielerlebnis wird oftmals durch die Verwendung zusätzlicher Dienste gestört. Die Nutzung zusätzlicher Informationen oder Funktionen, beispielsweise der Zugriff auf eine externe Wissensbasis, erfordert die Verwendung eigener Applikationen. Dies macht einen Taskwechsel zwischen dem Spiel und den verschiedenen Applikationen notwendig. Durch die Trennung von Spiel und Spielumfeld beschränkt sich das kreative Potenzial der Gaming-Community zum größten Teil auf das Umfeld des Spiels, da eine Integration in das Spiel heute in den meisten Fällen nicht möglich ist.

1.2 Ziele der Arbeit

Eine umfassende Unterstützung von Spielern innerhalb von virtuellen Welten trägt zu einer Verbesserung des Spielerlebnisses bei. Daher ist die primäre Zielsetzung dieser Arbeit, eine solche Unterstützung zu ermöglichen. Diese Unterstützung soll über die Verwendung externer Dienste, die außerhalb der virtuellen Welt zur Verfügung stehen und geeignet in die virtuelle Welt integriert werden, realisiert werden. Externe Dienste können zusätzliche Informationen und Funktionen bereitstellen, die in der virtuellen Welt selbst nicht zur Verfügung stehen.

Dabei ist ein weiteres Ziel, dass die Anbindung und Integration der Dienste nicht spezialisiert auf eine einzelne konkrete virtuelle Welt erfolgt. Die Erstellung von Diensten soll unabhängig von einer virtuellen Welt durchgeführt werden können und es soll möglich sein, Dienste in verschiedenen virtuellen Welten wiederzuverwenden. Auch die Möglichkeit spezialisierte Dienste einzusetzen, die eine gezielte Benutzerunterstützung angepasst auf spezielle Situation bieten, soll gegeben werden. Durch die Realisierung offener, einheitlicher Schnittstellen soll eine Beteiligung der Community bei der Entwicklung von Dienste, die dann in die virtuelle Welt integriert werden, ermöglicht werden. Ein weiteres Ziel der Arbeit ist, dass zusätzliche Dienste für einen Benutzer nur dann verfügbar sind, wenn sie auch nützlich sind. So wird ein Spieler nicht durch unnötige Dienste behindert. Um diese Zielsetzung zu erreichen, ist zum einen ein geeigneter Informationsaustausch zwischen einer virtuellen Welt und externen Diensten, die von verschiedenen Applikationen bereitgestellt werden können, zu ermöglichen und zum anderen ist eine Anpassung der Dienste und des Dienstangebots auf die Situation des Benutzers, seinen virtuellen Kontext, notwendig.

Die Ziele dieser Arbeit sind somit die theoretischen Grundlagen für eine Integration von Diensten in virtuelle Welten zu schaffen, ein geeignetes Konzept für einen generischen Informationsaustausch zwischen virtuellen Welten und externen Applikationen zu entwickeln sowie die Bereitstellung kontextbasierter Dienste zu ermöglichen und eine prototypische Implementierung als Proof-of-Concept zu realisieren.

1.3 Beiträge

Die vorliegende Arbeit schafft die Voraussetzungen und Grundlagen für die Integration von kontextbasierten Diensten in virtuelle Welten. Es wurde ein Informationsaustausch zwischen beliebigen virtuellen Welten und anderen Internet-Applikationen realisiert und die Verwendung von virtuellem Kontext ermöglicht. Auf Basis dieser Grundlagen wurden die Integration von externen Diensten zur Unterstützung von Spielern in virtuelle Welten und die Verwendung von virtuellem Kontext für Angebot und Steuerung der Dienste umgesetzt. Hierzu tragen folgende Aspekte der Arbeit bei:

- Basierend auf der im Rahmen der Arbeit durchgeführten umfangreiche Analyse virtueller Welten wurde ein Modell der Interaktion in virtuellen Welten entwickelt.
- Das Konzept des Kontexts, wie es im verwandten Gebiet der kontext-bewussten Dienste verwendet wird, wurde auf virtuelle Welten übertragen. Dazu wurden bestehende Definitionen von Kontext untersucht und die analysierten Eigenschaften virtueller Welten berücksichtigt.
- Um virtuellen Kontext technisch erfassbar und nutzbar zu machen, wurde mit der eXtensible Game Description Language (xgdl) eine Beschreibungssprache für virtuellen Kontext entwickelt. Grundlage von Erfassung und Verwendung des virtuellen Kontexts bilden die in der durchgeführten Analyse identifizierten virtuellen Parameter.
- Die Verwendung von virtuellem Kontext ermöglicht Angebot und Steuerung unterstützender Dienste in virtuellen Welten. Für die Umsetzung einer entsprechenden Benutzerunterstützung wurde das Konzept der virtuell kontextbasiert Dienste (Virtual Context Based Services - VCBS) eingeführt und definiert.
- Ein Konzept zur Verarbeitung virtueller Kontextinformationen für das Angebot und die Steuerung von virtuell kontextbasierten Diensten wurde entwickelt. Auf der Grundlage virtuell kontextbasierter Dienste wurde ein kontrollierter Informationsaustausch zwischen virtuellen Welten und externen Applikationen realisiert. Ein solcher Informationsaustausch ermöglicht die Verwendung von Informationen aus virtuellen Welten durch externe Dienste, sowie die Nutzung externer Zusatzdienste innerhalb von virtuelle Welten. Der Informationsaustausch umfasst die Akquise von virtuellen Kontextinformationen, deren Verteilung an externe Dienste, sowie die Integration der Dienstdaten in die virtuelle Welt.
- Es wurde eine Middleware-Architektur zur Realisierung des Informationsaustauschs und zur Bereitstellung kontextbasierter Dienste konzipiert (VCBS-Middleware). Die VCBS-Middleware beinhaltet Komponenten zur Erfassung von Kontextinformation in virtuellen Umgebungen, zur Kontextverarbeitung und Verteilung sowie zum Angebot von Diensten basierend auf der virtuellen Situation des Benutzers und zur Bereitstellung externer Dienste in der virtuellen Welt.
- Die VCBS-Middleware wurde prototypisch implementiert. Verschiedene Beispieldienste wurden konzipiert und implementiert sowie zwei Multiplayer Online Spiele, die exemplarisch für virtuelle Welten stehen, über die Middleware angebunden. Mit der VCBS-Middleware wird die Umsetzbarkeit des Konzepts der virtuell kontextbasierten Dienste nachgewiesen.
- Schließlich wurde das umgesetzte System hinsichtlich seiner Funktionalität evaluiert und es wurden Messungen zur Bestimmung ausgewählter Systemeigenschaften durchgeführt.

1.4 Struktur der Arbeit

Die vorliegende Arbeit gliedert sich in acht Kapitel. Nach der Einleitung erfolgt in Kapitel 2 eine Einführung in das der Arbeit zugrundeliegende Szenario, den Multiplayer Online Games, die eine besondere Form virtueller Welten darstellen. Auf Basis dieser szenariobezogenen Darstellung werden die Ziele der Arbeit detailliert herausgearbeitet. In Kapitel 3 werden die Grundlagen der Arbeit erläutert und die relevanten verwandten Arbeiten vorgestellt. Die verwandten Arbeiten gliedern sich in drei Teile: (1) Arbeiten aus dem Bereich der kontext-bewussten Systeme mit der Betrachtung von Kontextdefinitionen und der Realisierung kontext-bewusster Systeme, (2) Virtuelle Umgebungen, im speziellen Multiplayer Online Games und deren technische Grundlagen sowie (3) Online-Communities, insbesondere Gaming-Communities und deren Eigenschaften.

Die im Rahmen dieser Arbeit entwickelten theoretischen Grundlagen und Konzepte werden in Kapitel 4 vorgestellt. Es beinhaltet die wichtigsten Aspekte der im Rahmen der Arbeit durchgeführten Analyse

virtueller Welten, sowie die daraus abgeleiteten Ergebnisse: die virtuellen Parameter, das entwickelte generische Interaktionsmodell, die Definition und Beschreibung von virtuellem Kontext und die Definition der virtuell kontextbasierten Dienste. Basierend auf den theoretischen Erkenntnissen wird in Kapitel 5 das Konzept für die Realisierung eines Informationsaustauschs zwischen virtuellen Welten und externen Diensten auf Basis virtuell kontextbasierter Dienste vorgestellt, nachdem zunächst die Anforderungen für einen solchen Informationsaustausch vorgestellt und bestehende Ansätze bezüglich der Anforderungen bewertet wurden.

Die Implementierung des vorgestellten Konzepts wird in Kapitel 6 beschrieben. Neben der Beschreibung der Middleware-Implementierung werden die umgesetzte Plug-In basierte Anbindung virtueller Welten anhand von zwei Beispielen sowie die im Rahmen der Arbeit entwickelten und implementierten Dienste dargestellt. Die funktionale Evaluation sowie Messungen zu ausgewählten Systemparametern der implementierten Middleware werden in Kapitel 7 vorgestellt. Kapitel 8 schließt die Arbeit mit einer Zusammenfassung und einem Ausblick auf zukünftige Arbeiten ab.

2 Szenario

Das Szenario dieser Arbeit, in dem die Herausforderungen begründet sind und die Ergebnisse genutzt werden können, sind Multiplayer Online Games (MOGs) und die durch MOGs entstehenden Vernetzungen von Spiel und Spielern (*networked gaming*). MOGs sind aus zwei Gründen als Beispielszenario von Interesse. Sie sind momentan die erfolgreichsten und am weitesten verbreiteten virtuellen Welten und sie sind komplexe (soziale) Systeme, die sowohl einen Bedarf der Interaktion zwischen Benutzern hervorrufen, als auch einen Bedarf an Diensten zur Unterstützung der Benutzer innerhalb der virtuellen Umgebung. Die soziale Komponente der MOGs ist die Grundlage für die entstehende soziale Vernetzung der Spieler bis hin zur Bildung von Gaming-Communities.

Als Einführung in das Szenario und grundlegende Aspekte der Arbeit werden im Folgenden verschiedene Elemente von MOGs an einem Beispiel beschrieben.

2.1 Grundlagen von MOGs am Beispiel von *World of Warcraft*

Bis zu mehrere tausend Spieler spielen MOGs gemeinsam, mit- und gegeneinander. Es gibt verschiedene Typen von MOGs, beispielsweise First Person Shooter (FPS), Echtzeitstrategiespiele (RealTime Strategy, RTS) oder Massively Multiplayer Online RolePlaying Games (MMORPGs). Ein bekanntes Beispiel eines MMORPGs ist das Spiel *World of Warcraft* [11]. Dieses Spiel wird im Folgenden verwendet um verschiedene Eigenschaften von MOGs im Allgemeinen und MMORPGs im Speziellen zu erläutern. Weitere Informationen zu *World of Warcraft* finden sich beispielsweise auf der europäischen *World of Warcraft* Homepage [11] oder bei [Che09a].

Virtuelle Charaktere - Klassen, Eigenschaften und Fähigkeiten

World of Warcraft "spielt" in einem Fantasy Szenario. Jeder Benutzer muss einen virtuellen Charakter erstellen, um die virtuelle Welt des Spiels betreten zu können (siehe Abbildung 2.1). Die von den Spielern erstellen Charaktere unterscheiden sich in der Rasse und im Aussehen voneinander und haben unterschiedliche Klassen wie Magier, Priester oder Krieger.

Abbildung 2.1: Charaktererstellung in *World of Warcraft*

Nach der Charaktererstellung kann der Benutzer die virtuelle Welt mit seinem virtuellen Charakter betreten. Dieser besitzt verschiedene Eigenschaften und grundlegende Fähigkeiten. Die Eigenschaften eines virtuellen Charakters beinhalten beispielsweise verschiedene Attribute wie Intelligenz, Stärke, Ausdauer und Willenskraft oder dass der Charakter bestimmte Klassen von Gegenständen (zum Beispiel eine Plat-

tenrüstung) benutzen kann. Die grundlegenden Fähigkeiten eines Charakters beinhalten das Bewegen durch die virtuelle Umgebung, die Interaktion mit Nichtspieler-Charakteren (*Non Player Characters* - NPCs), wie Händler, Questgeber oder Lehrer, das Benutzen von virtuellen Objekten und das Durchführen von Aufgaben (*quests*). Zusätzlich beinhalten die Fähigkeiten des virtuellen Charakters bestimmte Fertigkeiten (*skills*) wie zum Beispiel Angeln, Feuerbälle werfen oder Schlösser knacken. Diese Fertigkeiten können je nach Charakterklasse und Rasse unterschiedlich sein. Neben den Aktionen, die ein virtueller Charakter aufgrund seiner Fähigkeiten ausführen kann, besitzt er Möglichkeiten mit anderen virtuellen Charakteren zu kommunizieren. Die Aktions- und Kommunikationsmöglichkeiten sind abhängig von der Definition der virtuellen Welt und ihrem Regelwerk und werden über das Benutzerinterface des Spiels bereitgestellt (siehe Abbildung 2.2).

Abbildung 2.2: Benutzerinterface von *World of Warcraft*

Durch das Sammeln von Erfahrungspunkten und Gegenständen und das Verbessern von Eigenschaften und Fertigkeiten entwickelt sich der Charakter weiter. Er kann nach einer Weile neue oder erweiterte Fertigkeiten lernen oder freischalten (beispielsweise über Talentbäume) und sich auf eine bestimmte Ausrichtung spezialisieren (sich beispielsweise als Magier zu einem Feuer- oder Eismagier entwickeln), wodurch er sein Repertoire an Fähigkeiten erweitert. Zusätzlich erhalten die Charaktere Ausrüstungsgegenstände wie Waffen, Rüstung und Schmuck. Die Verbesserung des Equipments (Ausrüstung) steigert dann wiederum verschiedene Eigenschaften oder Fähigkeiten. Auf ihrem Weg durch die virtuelle Welt kämpfen sie beispielsweise gegen Fantasiemonster die in Höhlen, Festungen oder Lagern in den abwechslungsreichen Landschaften des *Warcraft* Universums hausen.

Neben den Aufgaben (*Quest*), die das Spiel dem Benutzer stellt, bietet es eine Reihe weiterer Beschäftigungsmöglichkeiten, wie verschiedene Berufe, ein Handelssystem, Banken, zentrale Treffpunkte und verschiedene Chats. Für die Erfüllung von Aufgaben erhält der Spieler Erfahungspunkte und Gegenstände. Diese Gegenstände können entweder nützlich sein (wie ein Helm oder ein Heiltrank) oder es kann sich um Funitems (Spaßgegenstände) handeln, die keinen wirklichen Nutzen haben, aber zur Erheiterung beitragen.

Die virtuellen Charaktere entwickeln sich, durch die Verbesserung ihrer Eigenschaften und Fertigkeiten, bis sie das Levelmaximum (aktuell Level 80) erreicht haben. In *World of Warcraft* gibt es nicht nur ein Spielziel. Je nach Interesse und Einstellung sind unterschiedliche Ziele für einen Spieler relevant, wie beispielsweise das Erreichen der Höchststufe, der Wettkampf gegen andere Spieler (PvP), das Erreichen besonderer Errungenschaften, der Handel mit anderen Spielern oder die Sozialisierung mit anderen Spielern. Es gibt zusätzlich immer wieder kleinere Zwischenziele, wie etwa das Erlernen eines Reitskills (ein Reittier ermöglicht eine schnellere Fortbewegung), das Erkunden eines Gebietes oder das Verfolgen einer Questreihe. Nach Erreichen der Höchststufe fängt das Spiel eigentlich erst richtig an. Der sogenannte *Endgame content* (Spielinhalte für hochlevelige Spieler) bietet wieder neue Herausforderun-

gen und Ziele, wie zum Beispiel das Sammeln von hochwertigen Ausrüstungsgegenständen, spezielle PvP-Wettkämpfe oder das Erkunden und Meistern von bestimmten Instanzen (Instanzen sind spezielle Gebiete innerhalb der Spielwelt, die für jede Spielergruppe einzeln instantiiert werden).

Gruppenspiel

Aufgrund der Asymmetrie der Fähigkeiten von Charakteren aus verschiedenen Klassen, wird eine Kooperation sowohl im Kampf als auch in kurzen Begegnungen gefördert. Ein Magier hält im Vergleich zu einem Krieger viel weniger Schaden aus, kann jedoch mehr Schaden austeilen. Der Krieger besitzt Fertigkeiten um einen Gegner an sich zu binden, der Magier besitzt Fertigkeiten um einen Gegner auf Distanz zu halten. Vor allem für Spielinhalte im Endgame sind fest organisierte Gruppen notwendig, da viele Inhalte nur für Gruppen zur Verfügung stehen und vor allem die interessanten instantiierten Gebiete speziell für große Gruppen (*Raid* - Schlachtzüge mit 10-40 Spielern) ausgelegt sind und teilweise über mehrere Tage hinweg mit derselben Gruppe "durchgespielt" werden. Das Spiel bietet daher die Möglichkeit, Gilden zu bilden, in denen sich Spieler zusammenschließen können, um gemeinsam Aufgaben zu bearbeiten und gegen Gegner zu kämpfen.

Das Gruppenspiel hat einen besonderen Stellenwert nicht nur in *World of Warcraft*, sondern in allen MOGs. In MMORPGs wird das Gruppenspiel im Allgemeinen durch das Ungleichgewicht der verschiedenen Klassen gefördert und gefordert. Abhängig von seiner Klasse kann ein Spieler verschiedene Rollen innerhalb einer Gruppe übernehmen. Übliche Rollen in *World of Warcraft*, wie auch in anderen MMORPGs können in drei Kategorien unterteilt werden: *Tank* (Tank), *Damage Dealer* (Schadensausteiler) und *Supporter* (Unterstützer). Das Ziel einer Spielergruppe in einem virtuellen Kampf, egal ob gegen virtuelle Gegner (PvE) oder andere Spieler (PvP), ist der Sieg. Um dieses Ziel zu erreichen, müssen in der Regel die Lebenspunkte aller Gegner auf Null reduziert werden.

Tank: Der Tank steht dem Gegener direkt gegenüber (*first-line*). Seine Rolle ist es, die Gegner an sich zu binden und möglichst viel des gegnerischen Schadens auf sich zu ziehen. Dazu muss der Charakter, etwa durch eine entsprechende Rüstung und viele Lebenspunkte, möglichst viel Schaden aushalten und Fertigkeiten besitzen, um die Aufmerksamkeit (*Aggro*) der Gegner auf sich zu ziehen. Das Ziel des Tanks ist es dabei, dass die anderen Gruppenmitglieder so wenig Schaden wie möglich erhalten.

Damage Dealer (DD): Die Rolle eines DDs ist es, den Gegnern soviel Schaden wie möglich zuzufügen. Dazu muss der Charakter Fertigkeiten besitzen, die hohe dps (*damage per second* - Schaden pro Sekunde) ermöglichen. Eine weitere Unterteilung wird häufig vorgenommen in Nahkämpfer, die direkt am Gegener stehen und meistens physischen Schaden verursachen und Fernkämpfer, die entfernt stehen und meistens Zauberschaden verursachen. Diese Unterteilung ist jedoch nicht strikt, da es auch Ausnahmen gibt, wie etwa die Klasse der Fernkämpfer, die mit Pfeil und Bogen auch physischen Schaden verursachen. Das Ziel der DDs ist es, die Lebenspunkte der Gegner so schnell wie möglich auf Null zu reduzieren.

Supporter: Die Rolle eines Supportes ist es, den Rest der Gruppe bei der Durchführung ihrer Aufgaben zu unterstützen. Dazu muss der Charakter Fertigkeiten besitzen, die positiv auf die eigene Gruppe wirken, oder negativ auf die Gegner. Positive Effekte können zum Beispiel die Rüstung des Tanks verbessern oder den Schaden der DDs erhöhen. Negative Effekte für die Gegner können Unterbrechung oder Reduzierung von Fertigkeiten sein oder ein zeitweises aus dem Kampf nehmen von Gegnern (*Crowd Control*). Eine besondere Ausprägung der Supporter sind *Healer* (Heiler). Heilung ist die wichtigste Form an unterstützenden Funktionen, da Heiler dafür sorgen, dass die Lebenspunkte der Gruppenmitglieder größer Null bleiben und diese somit aktiv am Kampf teilnehmen können. Das Ziel der Supporter ist es, ohne direkten Schaden zu verursachen, den Rest der Gruppe bei der Erfüllung ihrer Rollen zu unterstützen.

Über diese spielbezogenen Rollen hinaus nehmen die Mitglieder einer Gruppe noch weitere Aufgaben wahr, die unter den Gruppenmitgliedern aufgeteilt werden. Dazu zählen beispielsweise die Koordination gemeinsamer Aktionen. Kooperation und Koordination zwischen Spielern sind somit wesentliche Erfolgskriterien. Die Mitglieder einer Gruppe benötigen Werkzeuge zur Unterstützung der Kooperation und Koordination. Nur dann können sie im Gruppenspiel erfolgreich sein.

Weitere Aktionen im Spiel

Kennzeichnend für das Spielen von *World of Warcraft* ist, dass jedem Spieler eine große Auswahl an Aktivitätsmöglichkeiten angeboten wird und er sich daraus, seinem Spielstil folgend, sein Spielerlebnis selbst "zusammenstellen" kann. Typischerweise besteht das Spielen aus einer Kombination von Berufstätigkeiten, Handel, sozialer Interaktion, Verwaltung der eigenen Gegenstände und dem "eigentlichen Spielen". Ein Charakter kann verschiedene Berufe beispielsweise den Abbau von Ressourcen in der Spielwelt oder das Herstellen von Gegenständen erlernen. Der Handel mit anderen Spielern umfasst das Kaufen und Verkaufen von Gegenständen und Dienstleistungen entweder direkt oder über ein spielinternes Auktionshaus. Die soziale Interaktion mit anderen Spieler findet innerhalb der Gilde und dem festen Freundeskreis (*Buddies*), mit Zweckgemeinschaften (*pick-up groups / random groups*) zur Erfüllung einer Quest oder für eine Instanz, oder auch beim zufälligen Aufeinandertreffen irgendwo in der Spielwelt statt. Die verschiedenen Gegenstände eines Charakters kann der Spieler in den Taschen des Charakters und in zusätzlichen Ablagefächern auf der Bank lagern und organisieren. Das "eigentliche Spielen", bei dem die klassenspezifischen Fähigkeiten eingesetzt werden, ist wiederum genauso facettenreich. Es beinhaltet Quests, Instanzen, Schlachtfelder (*Battlegrounds*), Arena-PvP, Erfolge, "Farmen" bis hin zu speziellen Weltereignissen (Spielinhalte, die nur für kurze Zeit zugänglich sind wie das Winterhauchfest zu Weihnachten, das Braufest im Oktober oder die Schlotternächte zu Halloween).

Aktivitäten im Umfeld des Spiels

Die genannten Aktionen führt ein Spieler innerhalb des Spiels aus. Außerhalb des Spiels setzen sich die spielbezogenen Aktionen fort (*Metagame*). Spieler suchen Informationen in diversen Foren, Wikis, Community-Portalen oder Webseiten. Sie dokumentieren ihre Erfahrungen, organisieren ihre Gruppe, kommunizieren mit anderen Spielern oder entwickeln eigene Tools oder Erweiterungen für ihr Spiel. Dadurch entstehen diverse Artefakte, die wiederum von den Spielern genutzt werden und unter anderem Hilfestellungen für alle möglichen Spielsituationen und Spielertypen bieten. Diese Artefakte beinhalten beispielsweise Wissenssammlungen, Guidelines, Forenbeiträge, Communityseiten, Spielmodifikationen, oder Kommunikationstools.

Zwei typische Beispiele für Aktivitäten, die in Bezug auf das Spiel durchgeführt werden, sind die Nutzung externer Sprachkommunikations-Tools und externer Informationssammlungen. Wie beschrieben, ist die Kooperation in der Gruppe ein wichtiges Spielelement, so dass vor allem feste Gruppen wie Spielergilden oft während des Spielens ein externes Sprachkommunikations-Tool (Voice-Chat) als unterstützenden Dienst einsetzen. Dafür wird zum einen ein spielexterner Voice-Chat-Server benötigt, den die Spielergruppe betreibt. Zum anderen benötigt jeder Spieler einen Voice-Chat-Client, der auf seinem System installiert ist. Durch die Verwendung des Voice-Chats sind Absprache und Koordination einer Gruppe viel einfacher und effizienter möglich, als beispielsweise mit dem im Spiel verfügbaren Text-Chat.

Externe Informationssammlungen enthalten diverse Informationen zu einem Spiel, wie beispielsweise ein Verzeichnis der im Spiel enthaltenen Gegenstände, Beschreibungen der Quests, Instanzenführer oder Guidelines zum Beispiel zu Talentverteilungen. Diese Informationen können den Spielern helfen, sich in der komplexen Spielwelt zurechtzufinden und sind somit ein wichtiges Element der Benutzerunterstützung.

Immersion und Unterbrechung des *GameFlow*

Ein Nachteil von externen Informationen und Funktionen ist, dass sie nicht direkt im Spiel verfügbar sind. Beispielsweise muss ein Spieler, der zusätzliche Informationen oder Guidelines zu einem Problem, das er gerade im Spiel hat, sucht, zuerst vom Spiel zu seinem Browser wechseln und eine Webseite oder ein Wiki mit den entsprechenden Informationen finden, diese Informationen abrufen und danach zum Spiel zurückkehren, um weiterspielen zu können. Das ist nicht nur umständlich, sondern auch eine immense Unterbrechung des Spiels.

In Untersuchungen wurde festgestellt, dass die Verwendung externer Tools die Immersion des Spielers und somit sein Spielerlebnis stören [Koi03]. Immersion ist ein wichtiger Faktor des Spielerlebnisses, be-

ziehungsweise des beim Spielen entstehenden "*GameFlow*" [SW05]. *Flow* beschreibt das Gefühl komplett in einer Aktivität zu versinken, mit einem hohen Grad an Vergnügen und Befriedigung [Deb02]. Das im *Flow* auftretende Ausblenden der Welt außerhalb des Spiels ist ein nicht zu unterschätzender Faktor des Gesamterlebnisses [Che07]. Mit der Identifikation des Spielers mit seiner Spielfigur wird das Erleben des Spiels (zum Beispiel, dass der Spieler die Erfolgserlebnisse des Charakters als seine eigenen wahrnimmt) erst möglich. Beim Spielen reicht es nicht aus, nur dazusitzen und zuzuschauen und eventuell irgendwelche kognitiven Schemata zu aktivieren. Der Spieler muss ein aktiver Teilnehmer werden. Wenn diese aktive Teilnahme erfolgreich ist, dann entsteht eine starke Spielerfahrung, die einen großen Einfluss auf die Aktionen und die Aufmerksamkeit des Spielers haben kann. Immersion ist grundlegender Bestandteil des Spielens [ME05]. Somit ist die Immersion ein essentieller Bestandteil des Spielerlebnisses, was wiederum bedeutet, dass das Eintauchen in die Spielwelt nicht durch den Wechsel zu externen Tools unterbrochen werden sollte, um ein kontinuierliches Spielerlebnis zu gewährleisten. Das Spiel kann auch nicht einfach durch den Benutzer angehalten werden, da die persistente Spielwelt kein Pausieren ermöglicht. Das Spielgeschehen geht weiter während der Benutzer einen Taskwechsel zu einer anderen Applikation durchführt. Dadurch kann der Benutzer nur an sicheren Plätzen in der virtuellen Welt (beispielsweise in einer Stadt) gefahrlos zu einer anderen Applikation wechseln, oder muss im schlimmsten Fall mit dem virtuellen Ableben seines Charakters rechnen, während er die externe Applikation nutzt.

Unübersichtlichkeit des Angebots an Informationen und Funktionen

Weitere Nachteile externer Tools und Informationen entstehen gerade aufgrund ihrer Menge und Vielfalt. Die Gaming-Communities sind auch im Umfeld der Spiele sehr aktiv. Sie verwenden eine Vielzahl verschiedener Tools für spielbezogene Aktionen und erstellen diverse Artefakte zu ihren Spielen. Das Angebot dieser spielexternen Informationen und Funktionen ist oft riesengroß und unübersichtlich. Es gibt dabei keine zentrale Instanz, die die verschiedenen Angebote zusammenfasst. Hilfreiche Informationen sind oft über verschiedene Stellen im Internet verteilt. Dadurch entfällt, vor allem für ambitionierte Spieler, ein beachtlicher Teil der Zeit, die sie in Zusammenhang mit ihrem Spiel verbringen, auf Aktivitäten im Umfeld des Spiels, inklusive der Suche nach benötigten Informationen.

Das Problem der Unübersichtlichkeit des Angebots zeigt sich beispielsweise bei den Addons für *World of Warcraft*. Addons sind kleine Spielerweiterungen, die in das Spiel integriert werden können, um beispielsweise das Aussehen des Benutzerinterfaces zu verändern. Da Addons von der Community erstellt werden können, gibt es tausende verschiedener Addons mit teilweise sehr ähnlicher, aber auch mit ganz unterschiedlicher Funktionalität. Das Auffinden und die Auswahl von geeigneten Addons wird dadurch für den Spieler zur Herausforderung. Es gibt keine zentrale Instanz, die alle verfügbaren Addons kennt, sodass verschiedene Quellen auf der Suche nach einem passenden Addon vom Benutzer durchsucht werden müssen. Auch gibt es für den Benutzer nicht die Möglichkeit, sich über neue Addons oder Aktualisierungen informieren zu lassen (wie beispielsweise bei *subscription-services*). Der Benutzer muss immer selbst tätig werden, sich also immer selbst auf dem Laufenden halten und zusätzliche Zeit investieren, um nach Neuerungen zu suchen. Ob ein gefundenes Addon wirklich für die Situation geeignet ist, muss der Benutzer letztendlich durch Ausprobieren herausfinden. Meist ist nur eine rudimentäre Beschreibung der Funktionalität vorhanden und es gibt keine gemeinsame Grundlage für solche Beschreibungen. Dadurch sind sie schwer vergleichbar. Dass es ein Addon für eine bestimmte Situation gibt, erfährt der Benutzer in der Regel über Freunde oder die Community. Oft würde ein Benutzer ein entsprechendes Addon benutzen wenn er wüsste, dass es existiert.

Die *World of Warcraft* Addons werden zwar in das Benutzerinterface des Spiels integriert, sind jedoch dann immer aktiv, auch wenn sie gerade nicht benötigt werden. Das führt im schlimmsten Fall dazu, dass die zusätzlichen Dienste, die durch die Addons bereitgestellt werden, den Benutzer nicht mehr unterstützen sondern eher behindern. Beispielsweise indem das eigentliche Spielgeschehen verdeckt wird, sodass der Benutzer nicht mehr direkt daran teilnehmen kann (siehe Abbildungen 2.3 und 2.4).

Abbildung 2.3: Beispiel für ein Benutzerinterface mit übermäßig vielen Addons [37]

Abbildung 2.4: Beispiel für ein Benutzerinterface mit übermäßig vielen Addons [37]

2.2 Zielsetzung

Das Ziel dieser Arbeit ist es, virtuellen Welten zu ermöglichen, Informationen aus der virtuellen Welt dem Umfeld bereitzustellen und Dienste aus dem Umfeld in die virtuelle Welt zu integrieren. Diese Dienste sollen innerhalb der virtuellen Welt, passend zu der virtuellen Situation des Benutzers (kontextbasiert) zur Verfügung gestellt werden. Durch die Realisierung eines solchen kontrollierten Informationsaustauschs zwischen virtuellen Welten und ihrem Umfeld können aktuell existierende Nachteile, wie der Verlust der Immersion durch Taskwechsel und die Unübersichtlichkeit des Angebots aufgrund der Informationsvielfalt, verhindert werden und eine aktive Beteiligung der Community ermöglicht werden.

Der Informationsaustausch zwischen MOGs und ihrem Umfeld ermöglicht, dass ein Spieler, ohne das Spiel zu verlassen, auf die Informationen, die die Gaming-Community in einer externen Applikation (etwa einer Web-Applikation) gesammelt hat, aus dem Spiel heraus zugreift. Ein unnötiger Taskwechsel wird somit vermieden und die Immersion des Spielers in der virtuellen Umgebung bleibt erhalten. Eine direkte Integration von Informationen und Funktionen in das Spiel-Interface erlaubt eine einheitliche Darstellung und eine einfache Bedienung. Die Spieler haben das Geschehen weiterhin im Auge, da sie nicht zu einem anderen Programm wechseln müssen. Darüber hinaus kann ein Spieler, in umgekehrter Richtung, seine Errungenschaften aus dem Spiel heraus in einem Community-Portal oder einer anderen Internet-Applikation darstellen und sich damit anderen Spielern präsentieren. Gerade für die Suche nach Mitspielern sind diese Informationen sehr wertvoll, da sich Gruppen aus Spielern mit verschiedenen Eigenschaften zusammensetzen und dadurch nur bestimmte Spieler für bestimmte Aufgaben in Frage kommen. Auch ist es für die anderen Spieler, die ein neues Gruppenmitglied suchen, sehr wichtig, dass die Informationen verläßlich sind. Dies ist bei Informationen gewährleistet, die direkt aus dem Spiel stammen, im Gegensatz zu Informationen, die der Spieler von Hand angibt. Darüber hinaus sind die verwendeten Informationen einheitlich und objektiv.

Damit das zusätzliche Angebot von Informationen und Funktionen im Spiel nicht unübersichtlich wird, ermöglicht der kontrollierte Informationsaustausch eine Steuerung der Bereitstellung anhand der Situation des Benutzers in der virtuellen Welt. Befindet sich der Spieler in dem MOG beispielsweise gerade in einem Kampf, dann sollten ihm keine Dienste angeboten werden, die ihn etwa beim Handel mit anderen Spielern unterstützen. Ist der Spieler jedoch gerade dabei eine bestimmte Aufgabe (*Quest*) zu lösen, dann könnte er beispielsweise die Möglichkeit angeboten bekommen, auf zusätzliche Informationen oder Hilfen zu seiner Quest zuzugreifen, die die Community außerhalb des Spiels gesammelt hat. Auch die Akquise von Informationen für die externe Informationssammlung kann dadurch direkt im Spiel durch-

geführt werden. Entdeckt ein Spieler beispielsweise etwas Interessantes, so hat er die Möglichkeit, seine Entdeckung direkt innerhalb des Spiels zu dokumentieren. Seine Angaben werden dann automatisch zusammen mit Informationen über seine aktuelle Situation im Spiel (Kontextinformationen) an die externe Informationssammmlung weitergegeben. Der nächste Spieler in derselben Situation kann dann diese Informationen wieder innerhalb des Spiels abrufen.

Zwischen dem MOG und seinem Umfeld soll somit ein Informationsaustausch realisiert werden, sodass das Spiel vom Engagement und dem kreativen Potenzial seiner Community profitieren kann und die Immersion der Spieler nicht durch unnötige Taskwechsel gestört wird. Das Angebot und die Bereitstellung der externen Informationen und Funktionen, also der zusätzlichen Dienste innerhalb der virtuellen Welt soll basierend auf dem Kontext des Benutzers an seine Situation angepasst werden, sodass keine Behinderung durch unnötigen Dienste entsteht und der Benutzer immer das aktuell relevante Angebot erhält und damit auch zum Beispiel über neue Dienste informiert wird.

3 Grundlagen

In diesem Kapitel werden die mit dieser Arbeit verwandten Forschungsbereiche vorgestellt und die grundlegenden Begrifflichkeiten erläutert und definiert. Im ersten Teil werden kontext-bewusste Systeme und deren Eigenschaften erläutert sowie Kontextdefinitionen dargestellt. Im zweiten Teil werden virtuelle Welten eingeführt und die im Fokus dieser Arbeit stehenden Multiplayer Online Games (MOGs) vorgestellt. Der dritte Teil schließt mit einer Betrachtung von Online- und Gaming-Communities das Grundlagenkapitel ab.

3.1 Kontext-bewusste Systeme

Die Situation einer Person kann durch ihren Kontext beschrieben werden. Jedoch gibt es keine eindeutige Definition von Kontext. In der Literatur finden sich eine Vielzahl verschiedener Definitionen. In der Informatik verwenden verschiedene Bereiche (wie z.B. Mensch-Maschine-Interaktion, Künstliche Intelligenz, Betriebssysteme oder Ubiquitous Computing) unterschiedliche Kontextbegriffe. Der für diese Arbeit relevante Kontextbegriff stammt aus dem Bereich der "*context-aware systems*" (siehe beispielsweise [BDR07]). Doch auch in diesem Forschungsbereich existiert eine Vielfalt an verschiedenen Kontextdefinitionen. Im Folgenden werden ausgewählte Definitionen aus verwandten Arbeiten vorgestellt und diskutiert.

Da es keine einheitliche Übersetzung von *context-aware* ins Deutsche gibt, wird in dieser Arbeit in Anlehnung an Görtz [Gör05] "kontext-bewusst" als deutsches Äquivalent verwendet.

3.1.1 Kontextdefinitionen

Im Bereich der kontext-bewussten Systeme gibt es drei grundlegende Ansätze für die Definition von Kontext: Die Verwendung von Synonymen und Umschreibungen, die Aufzählung von Beispielen und die Definition aus Systemsicht. Vor allem in den Kontextdefinitionen, die Ende der 90er Jahre entstanden sind, wurden synonyme Begriffe für Kontext verwendet um diesen zu definieren. Beispiele dafür sind Umgebung (*environment* z.B. in [WSA+96]), Situation (*situation* z.B. in [FF98]) oder Umfeld (*surroundings* z.B. in [WJH97]). Diese synonymen Begriffe ermöglichen eine abstrakte Kontextdefinition, die jedoch für eine Anwendung zu ungenau ist. Definitionen, die Synonyme verwenden, sind nur sehr schwer anwendbar, da die Synonyme meist ebenso vage wie der Begriff Kontext selbst sind und seine Bedeutung nicht fassbar darstellen.

Einen Schritt weiter gehen Hull et al. [HNBR97], die Kontext als Aspekte der lokalen Umwelt des Benutzers umschreiben. Somit ist Kontext eine unbegrenzte Menge von Aspekten bzw. Merkmalen. Diese Definition ist grundlegend. Sie zeigt, wie man Kontext handhabbar machen kann und dass es möglich ist, einzelne Merkmale von Kontext zu benennen. Der nicht begrenzten Anzahl von Merkmalen trägt Brown [Bro96] Rechnung, indem er Kontext als die Elemente der Umgebung des Benutzers definiert, welche dem Computer des Benutzers bekannt sind. Dieser Einschränkung unterliegen alle rechnerbasierten Systeme, die sich mit Kontext beschäftigen. Die Erfassung von Kontextinformationen für deren Verwendung in Computersystemen stellt immer noch eine Herausforderung dar.

Neben der Verwendung von Synonymen gibt es Kontextdefinitionen, die versuchen, Kontext anhand konkreter Beispiele zu definieren. Schilit und Theimer [ST94] bezeichnen Kontext als Lokation, nahe Personen und Objekte sowie Veränderungen dieser Objekte. Eine ganz ähnliche Definition verwenden auch Brown et al. [BBC97] die Lokation, Tageszeit, Jahreszeit, Temperatur, usw. als Aspekte des Benut-

zerkontexts definieren. In [DAW98] verwenden Dey et al. die Beispiele Lokation, emotionaler Zustand und soziales Umfeld des Benutzers, Tageszeit und Objekte im Raum. Diese Definitionen sind zwar intuitiv zu verstehen, jedoch nicht ausreichend eine allgemeingültige Definition von Kontext zu geben. Die Aufzählungen enthalten jeweils einige bestimmte Merkmale des Kontexts, die jedoch in den verschiedenen Definitionen variieren. Einen Konsens über die Kontextmerkmale für kontext-bewusste Systeme gibt es nicht. Da gerade der Kontext einer Person beliebig komplex sein kann, kann es in der realen Welt keine erschöpfende Aufzählung von Kontextmerkmalen geben.

Um die Kontextdefinition anhand von Beispielen zu abstrahieren, wurden Kategorisierungen von Kontextmerkmalen vorgeschlagen. Als Basis für eine solche Abstraktion benennen Schilit et al. [SAW94] die für den Bereich der kontext-bewussten Systeme wichtigen Aspekte: wo man ist, wer bei einem ist und welche Ressourcen in der Nähe sind. Diese Aspekte der sich ständig ändernden Ausführungsumgebung ("constantly changing execution environment" [SAW94]), bilden die Grundlage für die Unterteilung von Kontextmerkmalen in die Kategorien *Computerumgebung*, *Benutzerumgebung* und *physikalische Umgebung* (siehe auch [Dey00]).

- **Computerumgebung**: zur Verfügung stehende Prozessoren sowie Ein- und Ausgabegeräte, Konnektivität, ...
- **Benutzerumgebung**: Lokation, Personen in der Nähe, Soziale Situation, ...
- **Physikalische Umgebung**: Helligkeit, Geräuschpegel, Temperatur, ...

Eine Erweiterung dieser Kategorien schlagen Chen und Kotz [CK00] vor. Die Erweiterung beinhaltet eine zusätzlich Kategorie, den *Zeitkontext*, da der Aspekt der Zeit in keine der drei vorgeschlagenen Kategorien passt, jedoch wichtiger Bestandteil des Kontext ist.

- **Zeitkontext**: Tageszeit, Woche, Monat, Jahreszeit, ...

Definitionen, die einzelne Kontextmerkmale oder Klassen von Kontextmerkmalen (Kontextdimensionen [Sch09]) aufzählen, sind immer anwendungsspezifisch, da Kontext die ganze Situation beinhaltet und nicht nur einzelne Aspekte einer Situation. Es gibt eine unbegrenzte Anzahl von Kontextmerkmalen und die relevanten Merkmale können sich von Situation zu Situation ändern. Merkmale, die in manchen Fällen von großer Bedeutung sind, können in anderen Fällen unnötig sein und umgekehrt. Beispielsweise bezeichnet Dey [Dey00] vier Typen von Kontext als besonders relevant: **Lokation**, **Identität**, **Zeit** und **Aktivität**.

Basierend auf diesen Erkenntnissen formulierten Dey und Abowd [DA00] eine grundlegende Kontextdefinition die sehr weit verbreitet ist und genutzt wird:

> "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves.*"

Der Kontext beinhaltet demnach jede Information, die verwendet werden kann, um die Situation einer Entität zu beschreiben. Dabei kann eine Entität eine Person, ein Ort oder ein Objekt sein, welches relevant für die Interaktion zwischen einem Benutzer und einer Applikation ist, inklusive des Benutzers und der Applikation selbst. Diese Definition ermöglicht zum einen die Abstraktion von einer konkreten Anwendung (im Gegensatz zu den aufzählenden Definitionen), gibt dafür aber konkrete Angaben wie Kontext beschrieben werden kann (im Gegensatz zu den Synonymen). Dabei sind *Kontextmerkmale* oder *Kontextinformationen* relevante Informationen, die zur Charakterisierung eines Kontexts in einem bestimmten Gültigkeitsbereich verwendet werden können [Gör05].

Die Kontextdefinition von Dey und Abowd wird in fast allen späteren Arbeiten als Grundlage benutzt und meist auf die jeweilige Problemstellung hin angepasst verwendet. So auch zum Beispiel bei Görtz [Gör05], der zusätzlich in seiner Definition noch einmal hervorhebt, dass die Beschreibung sowohl abstrakt als auch bedeutsam sein soll:

*"Ein **Kontext** ξ ist eine abstrakte, bedeutungstragende Beschreibung einer Aktion eines Objektes und ihrer Beziehung zu anderen Objekten. Ein Objekt kann dabei eine Situation, eine Entität oder eine Umgebung sein."*

Eine etwas andere Herangehensweise kennzeichnen Kontextdefinitionen aus Systemsicht. Dort wird die Anzahl der verwendeten Merkmale beschränkt, indem nur noch die Kontextinformationen berücksichtigt werden, welche Relevanz für eine bestimmte Applikation haben. Es wird gar nicht erst versucht, Kontext mit all seinen Merkmalen zu beschreiben, sondern es werden nur diejenigen Kontextinformationen betrachtet, die für eine bestimmte Applikation von Bedeutung sind. So definieren Kofod-Petersen und Mikalsen [KPM05] Kontext aus Sicht einer kontext-bewussten Applikation:

"Context is the set of suitable environmental states and settings concerning a user, which are relevant for a situation sensitive application in the process of adapting the services and information offered to the user."

3.1.2 Definition kontext-bewusster Systeme

Benutzt man Kontextinformationen in Applikationen, die mit einem Benutzer interagieren, dann werden diese Applikationen als "kontext-bewusste Systeme" bezeichnet (*context-aware systems*). Analog dazu werden in der Literatur auch andere Begriffe wie *context-sensitive*, *situation aware* oder *situation sensitive* verwendet, wobei der Begriff *context-aware* am weitesten verbreitet ist und sich in aktuellerer Literatur durchgesetzt hat. Der Begriff "*context-aware*" wurde zuerst von Schilit und Theimer [ST94] eingeführt (1994). Die ersten Ansätze aus diesem Bereich reichen jedoch zurück bis zum Active Badge System von Olivetti Research (1992) [WFG92] und Active Map von Xerox PARC (1993) [Wei93], [WSA+96]. Über Begrifflichkeiten und den Begriff "*context-aware*" finden sich in der Literatur zahlreiche Diskussionen (siehe z.B. Dey und Abowed [DA00]).

Eine Applikation, die sich des Kontexts ihres Benutzers bewusst ist, kann auf Änderungen der Situation des Benutzers reagieren. Dabei kann eine Applikation Kontext benutzen, etwa, um zum Kontext passende Informationen anzuzeigen oder sich an den Kontext anpassen, also je nach Kontext unterschiedliche Funktionalitäten bereitstellen (siehe zum Beispiel [SAW94], [BBC97], [ADOB98]). Viele Ansätze verstehen Applikationen, die von kontext-bewussten Systemen bereitgestellt werden, als Applikationen, die sich bezüglich des Benutzerkontexts dynamisch ändern oder anpassen. Dey und Abowd definieren kontext-bewusste Systeme als Systeme, die dem Benutzer, auf Basis von Kontextinformationen, Informationen und/oder Dienste anbieten, die nützlich für seine Aufgabe sind:

"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

Eine erweiterte Unterteilung von Kontextinformationen unterscheidet zwischen aktivem und passivem Kontext. Der aktive Kontext beeinflusst das Verhalten einer kontext-bewussten Applikation direkt. Der passive Kontext ändert das Verhalten der Applikation hingegen nicht, nur die angezeigten oder gespeicherten Informationen werden angepasst. Jede Kontextinformation kann sowohl aktiver als auch passiver Kontext sein, da diese Unterscheidung anwendungsspezifisch ist ([CK00]).

Es gibt also die unterschiedlichsten Kontextdefinitionen, in verschiedenen Abstraktionsstufen und mit variierender Anwendbarkeit. Einen Überblick über die Ausprägungen der Kontext-Definitionen gibt Abbildung 3.1.

Darüber hinaus gibt es verschiedene Ansätze, Kontextinformationen zu kategorisieren. Verschiedene Autoren (wie [Gwi00]) verwenden die Unterteilung in Kontextdimensionen. Dabei wird unterschieden zwischen Kontextinformationen, die den Zustand der Umgebung beschreiben und durch Sensoren erfasst werden können (externer oder physikalischer Kontext) und solchen, die den Zustand des Benutzers beschreiben (wie Erfahrungen, Aufgaben oder Ziele) und in der Regel durch den Benutzer spezifiziert

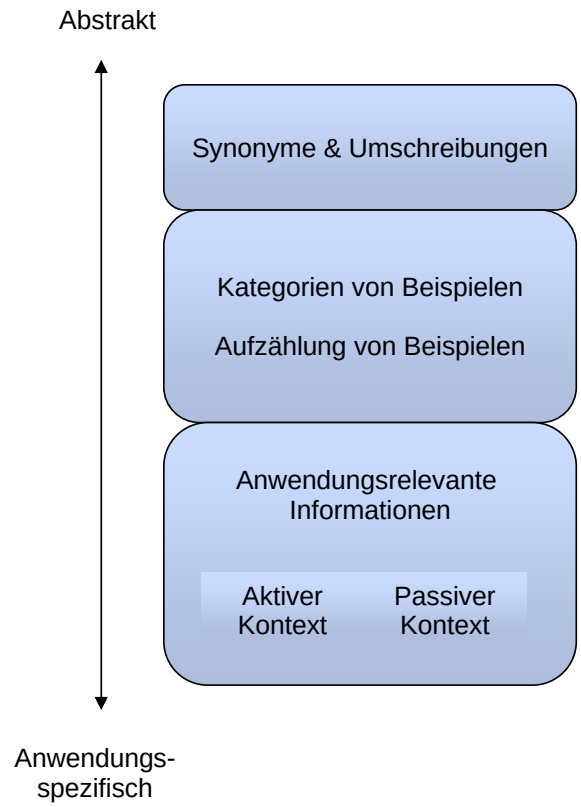

Abbildung 3.1: Überblick: Ausprägungen von Kontextdefinitionen

werden müssen (interner oder logischer Kontext). Neben der Erfassung und Verwendung einzelner Kontextmerkmale gibt es weiterführende Bestrebungen, erweiterte oder abgeleitete Kontextinformationen zu erzeugen [Dey01]. In diesen Fällen wird dann von sekundären Kontextinformationen gesprochen, da diese Informationen durch eine Kombination und / oder Interpretation von Kontextinformationen entstehen, im Gegensatz zu den primären Kontextinformationen, die direkt erfasst werden. Zur Erzeugung erweiterter Kontextinformationen können beispielsweise Ontologien ([PVdBW+04]) oder andere semantische Werkzeuge genutzt werden.

3.1.3 Eigenschaften kontext-bewusster Systeme

In der Literatur werden verschiedene Kategorisierungen verwendet um kontext-bewusste Systeme einordnen und vergleichen zu können. Pascoe [Pas98] kategorisiert kontext-bewusste Systeme anhand verschiedener Funktionalitäten, wie Erfassung, Anpassung, Auffinden von Ressourcen und Anreicherung.

Eine verbreitete Kategorisierung wurde von Schilit et al. [SAW94] vorgenommen. Sie unterscheiden vier Kategorien von Applikationen, die durch kontext-bewusste Systeme bereitgestellt werden können:

- **Auswahl nahe gelegener Objekte** (*proximate selection*) und **kontextabhängige Informationen** (*contextual information*): Bei der Auswahl nahe gelegener Objekte werden dem Benutzer Informationen darüber angezeigt, welche Objekte sich in seiner Nähe befinden (z.B. Drucker in meiner Nähe). Der Benutzer soll somit in seiner Entscheidung unterstützt werden, welches dieser Objekte er verwenden möchte. Er ist jedoch derjenige, der aktiv die Auswahl trifft. Bei kontextabhängigen Informationen wird die Sicht auf Informationen in den vom Benutzer verwendeten Applikationen angepasst.

- **Automatische kontextabhängige Rekonfiguration** (*automatic contextual reconfiguration*): Automatische Rekonfiguration bedeutet, dass sich die Applikation automatisch an Kontextänderungen anpasst und sich dadurch beispielsweise zugreifbare Informationen je nach Kontext ändern oder Ressourcen auf nahe gelegenen Computersystem genutzt werden können.

- **Kontextabhängige Informationen und Befehle** (*contextual information and commands*): Kontextabhängige Befehle können entweder nur in einem bestimmten Kontext ausgeführt werden oder aber anders parametrisiert ausgeführt werden. Beispielsweise wird ein kontextabhängiger "Druck"-Befehl automatisch auf dem nächstliegenden Drucker ausgeführt.

- **Durch Kontext ausgelöste Aktionen** (*context-triggered actions*): Durch Kontext ausgelöste Aktionen benötigen im Gegensatz zu den kontextabhängigen Befehlen keine Benutzereingabe, sondern werden automatisch aktiviert, basierend auf vordefinierten IF-THEN Regeln, hierzu zählt beispielsweise ein automatisches Einschalten der Beleuchtung, wenn eine Person im Raum ist.

Schilit et al. ordnen diese Kategorien anhand zweier Dimensionen (siehe Abbildung 3.2). Sie unterscheiden zwischen dem Informationserwerb (*information*) und der Ausführung einer Anweisung (*command*) sowie, ob der Benutzer selbst aktiv werden muss (*manual*) oder ob die Ausführung automatisch druchgeführt wird (*automatic*).

	manual	automatic
information	proximate selection & contextual information	automatic contextual reconfiguration
command	contextual commands	context-triggered actions

Abbildung 3.2: Dimensionen kontext-bewusster Systeme [SAW94]

Dey, Aboud und Salber [DA00], [Sal00] entwickeln diese Kategorisierungsansätze weiter und definieren ihrerseits drei Arten von kontext-bewussten Applikationen:

- ***Darstellung* von Informationen und Diensten** (*presentation of information and services to a user*) umfasst die Anzeige von Informationen und Diensten, die im aktuellen Kontext relevant sind.

- **Automatische *Ausführung* eines Dienstes** (*automatic execution of a service*) beinhaltet sowohl die automatisierte Ausführung eines Dienstes wie auch die automatisierte Anpassung eines Dienstes.

- ***Tagging* von Informationen** (*tagging of context to information for later retrieval*) bedeutet, dass Informationen automatisch mit dem aktuellen Kontext des Benutzers verknüpft werden (der Benutzer kann z.B. eine Notiz hinterlassen, die dann mit seinem Kontext verknüpft wird und in diesem Zusammenhang später wieder abgerufen werden kann).

Die verschiedenen Kategorisierungen aus der Literatur wurden im Rahmen dieser Arbeit abstrahiert und zusammengefasst, um einen Überblick über das Verhältnis von Automatisierung und Benutzerinteraktion in den verschiedenen Kategorien zu geben (siehe Abbildung 3.3). Gemeinsame Grundlage der Kategorisierungen ist, wie der von einem kontext-bewussten System erfasste Benutzerkontext in den von den Applikationen des Systems bereitgestellten Diensten (*kontext-bewusste Dienste*) verwendet wird. Das wichtigstes Merkmal für die Unterscheidung der verschiedenen Kategorien ist der Grad der Benutzerinteraktion der kontext-bewussten Dienste. Der Benutzer eines kontext-bewussten Systems kann in unterschiedlichem Maße das Verhalten des Systems und der Dienste mitbestimmen. Dem steht der Grad der Automatisierung der bereitgestellten Dienste entgegen. Zum einen kann ein Dienst automatisch ausgeführt werden, falls der Dienst zum dem aktuellen Kontext eines Benutzers passt. Zum zweiten kann ein Dienst automatisch angepasst werden, falls sich der Kontext eines Benutzer ändert.

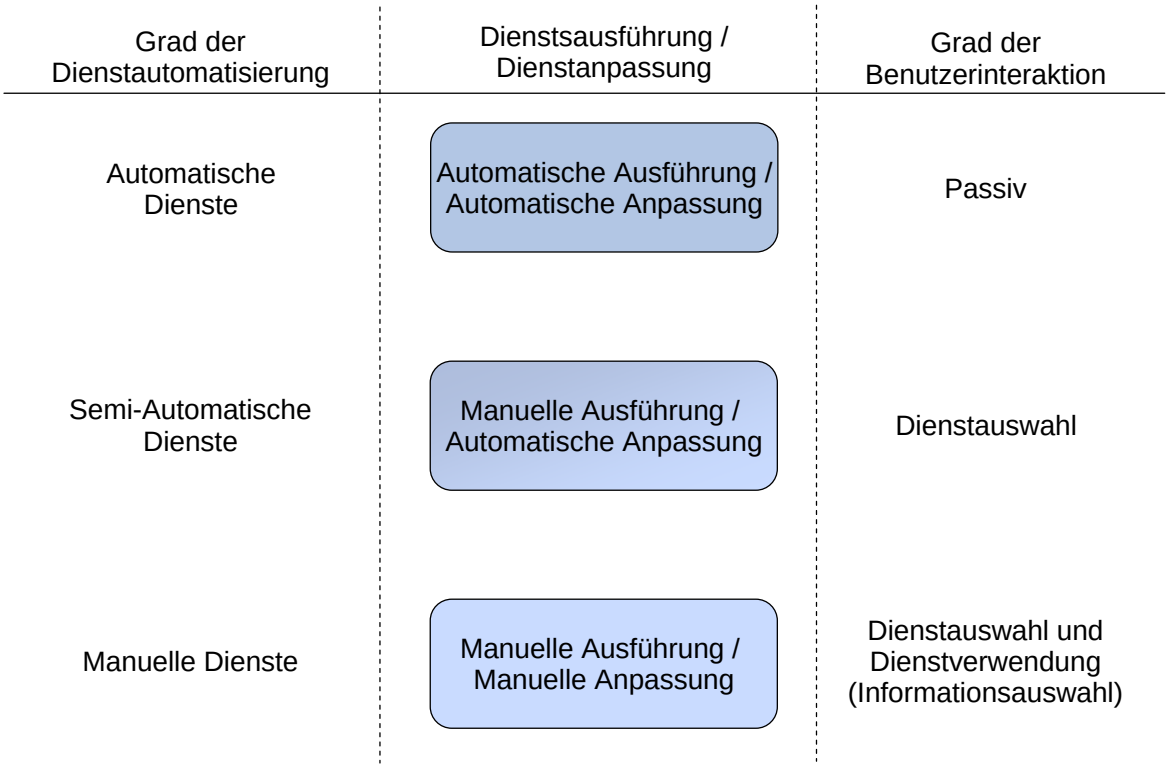

Abbildung 3.3: Überblick über das Verhältnis von Grad der Automatisierung und Grad der Benutzerinteraktion

Die oberste Kategorie, **automatische Dienste**, beinhaltet Dienste, die, sobald sie zu dem aktuellen Benutzerkontext passen, automatisch ausgeführt und, wenn sich der Benutzerkontext ändert, automatisch angepasst werden. Der Benutzer ist bei diesen Diensten passiv, d.h. er ist nicht in der Lage mitzubestimmen, wie sich der Dienst verhalten soll. Dafür muss der Benutzer aber auch nicht immer erst alles bestätigen, um es nutzen zu können, und wird somit auch nicht unterbrochen (keine Benutzerinteraktion). Beispiele für automatische Dienste finden sich etwa in smarten Räumen, die automatisch das Licht einschalten, wenn ein Benutzer den Raum betritt (*automatische Ausführung*) und die Helligkeit der Beleuchtung je nach Benutzeraktion ändern, z.B. das Licht dimmen wenn eine Präsentation gehalten wird (*automatische Anpassung*). Dienste dieser Kategorie werden auch als aktiv kontext-bewusst bezeichnet. Die unterste Kategorie enthält die **manuellen Dienste**. Im Falle von manuellen Diensten werden dem Benutzer zum einen die in seinem aktuellen Kontext verfügbare Dienste angeboten (Dienstauswahl) und der Benutzer muss diese Dienste aktivieren, um sie nutzen zu können. Zum anderen werden dem Benutzer zusätzlich die in seinem Kontext verfügbaren Dienstleistungen (Informationen, etc.) der genutzten Dienste (Dienstverwendung bzw. Informationsauswahl) angeboten. Auch die angebotenen Dienstleistungen (Informationen, etc.) muss der Benutzer auswählen, um sie verwenden zu können. Dadurch hat der Benutzer eine umfassende Kontrolle, ist aber auch stark gefordert (hoher Grad der Benutzerinteraktion). Beispiele für manuelle Dienste sind lokationsbasierte Touristenguides, die Informationen zu historischen Gebäuden oder Museen anbieten. Mit einem entsprechenden mobilen Gerät kann ein Benutzer diesen Dienst aktivieren (*manuelle Ausführung*) und, falls ihn ein Exponat interessiert, die dazu (auf Basis seines aktuellen Kontexts) angebotenen Informationen abrufen (*manuelle Anpassung*). Dienste dieser Kategorie werden auch als passiv kontext-bewusst bezeichnet. Dazwischen gibt es noch eine weitere Option, die **semi-automatischen Dienste** in der mittleren Kategorie. Bei semi-automatischen Diensten wählt der Benutzer die Dienste, die er nutzen möchte. Diese passen sich dann jedoch automatisch an (mittlerer Grad der Benutzerinteraktion). Diese Stufe stellt einen guten Kompromiss zwischen Mitbestimmung und Unterbrechung des Benutzers dar. Für das Beispiel des Touristenguides bedeutet das, dass ein Benutzer zwar den Dienst manuell aktivieren muss, um ihn nutzen zu können (*manuelle Ausführung*), nach der

Aktivierung dann jedoch, entsprechend der aktuellen Lokation des Benutzers, automatisch die jeweils passenden Informationen angezeigt werden (*automatische Anpassung*), und der Benutzer diese bei Interesse nicht erst noch abrufen muss. Eine weitere Möglichkeit ist die **Personalisierung von Diensten**. Bei der Personalisierung spezifiziert der Benutzer vorab das Verhalten der Dienste in bestimmten Situationen. Das setzt jedoch eine aufwändige Auseinandersetzung des Benutzers mit den Diensten voraus, noch bevor er diese überhaupt nutzen kann.

Welcher Kategorie ein kontext-bewusster Dienst angehört, wird bei der Erstellung des Dienstes festgelegt und ist abhängig vom kontext-bewussten System und der Applikation, die den Dienst bereitstellt. Die Kategorie eines Dienstes sollte je nach Anwendungskontext gewählt werden. Da dem Benutzer durch die Automatisierung ein Teil der Kontrolle über die Applikation genommen wird, ist es wichtig für die Akzeptanz solcher Applikationen, ein entsprechendes Maß an Interaktivität und Nützlichkeit für den Benutzer zu bieten [BD03].

3.1.4 Realisierung kontext-bewusster Systeme

Die Bereitstellung kontext-bewusster Systeme wirft verschiedene Forschungsfragestellungen auf, und ihre Realisierung ist mit verschiedenen Herausforderungen verbunden. Manche dieser Fragestellungen sind abhängig vom Anwendungsszenario; andere sind allgemeingültig in allen Bereichen, in denen kontext-bewusste Dienste eingesetzt werden. Die Entwicklung entsprechender kontext-bewusster Systeme ist komplex [dRE06] und es gibt keine generische Referenz, die für die unterschiedlichen Anwendungsbereiche abstrahiert ist und effizient eingesetzt werden kann.

In der Literatur werden vielfältige Forschungsfragestellungen und Herausforderungen bei der Realisierung kontext-bewusster Systeme diskutiert, einen Überblick geben die folgenden Beispiele. In Pauty et al. [PPRB06] werden verschiedene Herausforderungen diskutiert, die bei der Entwicklung von mobilen kontext-bewussten Diensten entstehen. Diese umfassen verschiedene Aspekte, etwa die Verteilung von Kontextinformationen oder die Beschreibung von Diensten. Darüber hinaus stellen die Autoren fest, dass die Anpassung von Diensten eine ungelöste Forschungsfrage darstellt und eine generische Beschreibungssprache benötigt wird. Gu et al. [GPZ05] adressieren Herausforderungen, wie einheitliche Kontextmodelle für Einheiten und Dienste oder Methoden für verschiedene Aufgaben wie die Akquise, die Verarbeitung oder die Verteilung von Kontextinformationen. Görtz [Gör05] betrachtet vor allem Erfassung, Verarbeitung (Synthese), Verteilung und Nutzung von Kontextinformationen. Als weitere Herausforderungen werden die Aspekte Sicherheit und Privacy diskutiert (siehe [CFJ04]).

Zusammengefasst kann man feststellen, dass kontext-bewusste Systeme verschiedene Anforderungen erfüllen und eine Reihe von Funktionen bereitstellen müssen. Die Funktionen werden in der Literatur unterschiedlich bezeichnet, lassen sich aber in fünf Bereiche gliedern. Abbildung 3.4 zeigt die fünf Phasen der kontextrelevanten Funktionen: *Kontextbeschreibung*, *Kontexterfassung*, *Kontextverarbeitung*, *Kontextverteilung* und *Kontextnutzung*.

Abbildung 3.4: Bereiche kontextrelevanter Funktionen

Diesen fünf Phasen lassen sich die einzelnen Funktionen zuordnen. Eine spezielle Phase ist die Kontextbeschreibung oder auch Kontext-Repräsentation, da sie die Grundlage für die weiteren Phasen bildet. Die Kontextbeschreibung muss nur einmal definiert werden. In der Phase (1) werden die Kontextinformationen gesammelt (*Discovery*, Akquise, Sammlung); darauf folgt in Phase (2) die Verarbeitung der gesammelten Informationen (*Processing*, Aggregation, Interpretation). In Phase (3) werden die Kontex-

tinformationen dann weitergegeben (*Dissemination*, Verteilung, Teilen) und in der letzten Phase (4) genutzt. Dabei müssen in jeder Phase Datenschutz- und Sicherheitsaspekte beachtet werden (*Privacy protection*).

Architekturen kontext-bewusster Systeme

Bei der Erstellung von kontext-bewussten Systemen kommen unterschiedliche Architekturen zur Anwendung. Baldauf et. al. [BDR07] geben einen Überblick über verschiedene Architekturen, die bei kontext-bewussten System eingesetzt werden. Als Grundlage für die Auswahl der passenden Architektur benennen sie die Umstände, in denen eine Applikation eingesetzt werden soll (wie Sensoren, Benutzeranzahl, Endgeräte). Dabei ist die Kontextaquise einer der wichtigsten Faktoren. Chen [Che04] unterscheidet drei Ansätze basierend auf der Art der Kontextakquise. Die Kontextakquise wird typischerweise über Sensoren realisiert. Diese Sensoren können entweder Hardware-Sensoren (Temperatursensor, etc.) oder Software-Sensoren (Prozess-Sensor, etc.) sein. Die drei Ansätze sind folgende:

Direkter Zugriff auf Hardware Sensoren: Dieser Ansatz kann nur verwendet werden, wenn Sensoren und Service auf einem Gerät integriert sind, so dass der Dienst bei Bedarf direkt auf die Sensordaten zugreifen kann. Er wird für mobile Endgeräte verwendet, ist jedoch sehr limitiert und nur relevant, wenn es sich um wenige Sensoren und einzelne Dienste handelt.

Middleware-Infrastruktur: Der Middleware-Ansatz führt verschiedene Layer ein, sodass die Kontextakquise gekapselt wird. Middleware-Infrastrukturen existieren in verschiedenen Ausprägungen. Typischerweise werden Middleware und Services auf demselben Gerät untergebracht. Die Services werden dann in Form von Agenten realisiert, die die Middleware anfragen können, oder es werden von der Middleware Widgets [Dey00] bereitgestellt, die den Zugriff auf bestimmte Kontextinformation ermöglichen.

Kontext-Server: Der Kontext-Server stellt Kontextinformationen für verschiedene Services bereit. Dadurch kann die Kontextakquise von einem ressourcenstarken Gerät durchgeführt werden und die Services können sich auf einem ressourcenschwachen Endgerät befinden und über den Kontext-Server die Kontextinformationen abfragen.

In verteilten Umgebungen sind Ansätze relevant, die auf einer "Middleware-Infrastruktur" oder einem "Kontext-Server" basieren. Verschiedenste Systeme wurden in diesem Bereich entwickelt, die sich beispielsweise im Funktionsumfang, oder in der Bezeichnung der Komponenten unterschieden. Im Folgenden werden ausgewählte Ansätze und Implementierungen kontext-bewusster Middleware-Architekturen betrachtet, die exemplarisch für eine ganze Reihe von Arbeiten stehen.

Mit dem **Context Toolkit** stellen Dey et al. [DAS01] ein konzeptuelles Framework vor und basierend darauf ein Toolkit für die Entwicklung kontext-bewusster Systeme. Das konzeptuelle Framework beinhaltet verschiedene Komponenten für die unterschiedlichen Funktionen kontext-bewusster Systeme. Es wird durch das Context Toolkit instanziiert (siehe Abbildung 3.5). Die Komponenten des Frameworks sind *Context Widgets*, die die Kontextakquise kapseln, *Interpreter* für die Interpretation von Kontextinformationen (wie die Assoziation geographische Koordinaten mit bestimmten Räumen), *Aggregatoren* für die Kombination einzelner Kontextinformation (wie etwa aller Kontextinformationen, die sich auf einen bestimmten Benutzer beziehen), *Services* welche Aktionen ausführen (wie das Licht ein oder ausschalten) und *Discoverer*, die das Framework verwalten und über die verfügbaren Komponenten Bescheid wissen. Applikationen können über einen *Discoverer* die Komponenten des Frameworks erfragen und diese dann direkt nutzen. Dafür kommunizieren sie direkt mit den einzelnen Komponenten, wobei die einzelnen Komponenten auch miteinander kommunizieren. Das Context Toolkit verwendet HTTP als Netzwerk Protokoll und XML als Datenformat, um die Interoperabilität verschiedener Systeme zu ermöglichen.

Gu et al. [GPZ05] haben ihrerseits eine service-orientierte kontext-bewusste Middleware namens **SOCAM** (*Service-Oriented Context-Aware Middleware*) entwickelt. Ihre Architektur besteht aus einer verteilten Middleware für *Ubiquitous Computing*. Die SOCAM Architektur ist unterteilt in drei Layer, das *Context Sensing Layer* also die Sensoren, das *Context Middleware Layer* und das *Context Application Layer*. Das

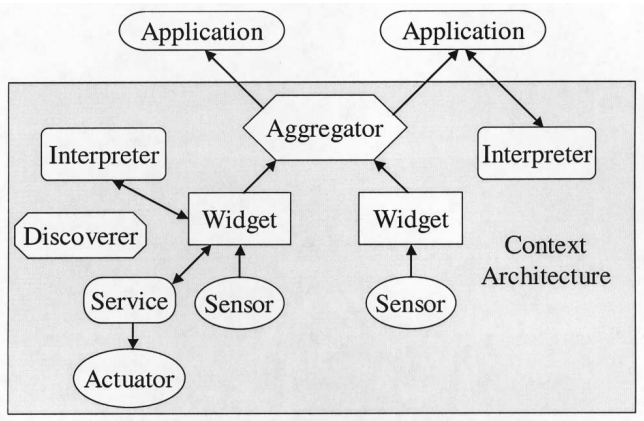

Abbildung 3.5: Context Toolkit Komponenten [DAS01]

Middleware-Layer besteht aus folgenden Komponenten, die unabhängig voneinander agieren (siehe Abbildung 3.6): *Context Provider* akquirieren und aggregieren Kontextinformationen, *Context Interpreter* führen eine Interpretation der Kontextinformation durch, eine *Context Database* speichert sowohl die zugrundeliegende Kontext-Ontologien als auch alte Kontextinformationen und der *Service Locating Service* verwaltet die internen *Middleware-Services*. Diese internen Middleware-Services beinhalten die Komponenten *Context Provider* und *Context Interpreter*, welche Kontextinformationen bereitstellen können. Für die Kommunikation der verschiedenen verteilten Komponenten der SOCAM Architektur wird Java RMI (*Remote Method Invocation*) eingesetzt. Die eigentlichen kontext-bewussten Dienste befinden sich im *Application-Layer* und können die Kontextinformationen der Middleware nutzen.

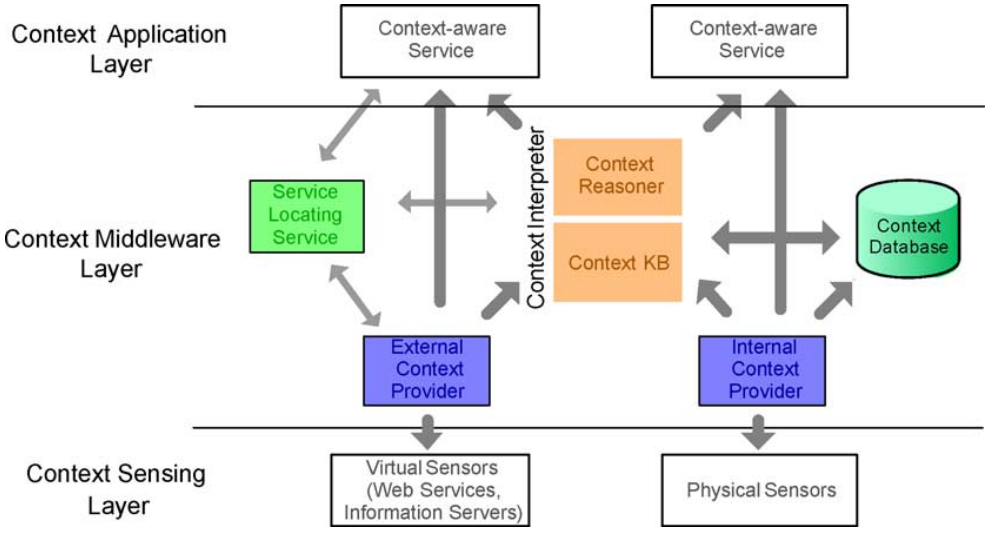

Abbildung 3.6: SOCAM Architektur [GPZ05]

Gaia [RHC+02] ist eine Middleware-Infrastruktur für smarte Räume (*active spaces*). Mit dem Meta-Betriebssystem *Gaia OS*, das die Heterogenität smarter Räume abstrahiert, können mobile benutzerbezogene Applikationen bereitgestellt werden. Gaia verwaltet die Ressourcen eines smarten Raumes und bietet Funktionalitäten für Lokalisierung, Kontext, Ereignisse und ein Repository mit Informationen über den smarten Raum an. Die Hauptbestandteile von Gaia sind der *Gaia Kernel*, das *Gaia Application Framework* und die *Active Space Applications*. Der Gaia Kernel beinhaltet das Management der verteilten Komponenten und kapselt die Basisfunktionalitäten wie den Event-Manager. Das Gaia Application Fra-

mework unterstützt Applikationen für smarte Räume. Die Gaia-Komponenten sind verteilte Objekte, für die Kommunikation und Interaktion dieser Objekte wird CORBA [46] eingesetzt.

CoBrA, die *Context Broker Architektur* [CFJ04], ist eine agentenbasierte Architektur für smarte Räume. Das zentrale Element ist der *Context Broker*, der ein Kontextmodell bereitstellt, Kontextinformationen akquiriert und interpretiert und die Einhaltung von Privacy-Policies der Benutzer überwacht (siehe Abbildung 3.7). Das System basiert auf der Verwendung von Ontologien (COBRA-ONT) und besteht aus vier Komponenten. Die *Context Knowledge Base* für die Persistierung von Kontextinformationen, die *Context Reasoning Engine*, die über das gespeicherte interferiert, das *Context Acquisition Module*, das wie die Context Widgets die Akquise der Kontextinformationen kapselt und das *Policy Management Module*, welches die Benutzer-Policies durchsetzt. Die Aufgaben des *Context Brokers* beinhalten, beim Austausch von Informationen die vom Benutzer gewünschte Privatsphäre einzuhalten und nur solche Informationen mit seinen Agenten auszutauschen, die vom Benutzer freigegeben wurden. CoBrA verwendet zu diesem Zweck einen komplexen policybasierten Ansatz, wobei die Policies mittels Ontologien definiert werden. Policies können dabei als individuelle Regeln eines Benutzers definiert werden, die bestimmen, ob eine gewisse Entität des Systems eine Aktion ausführen darf oder nicht.

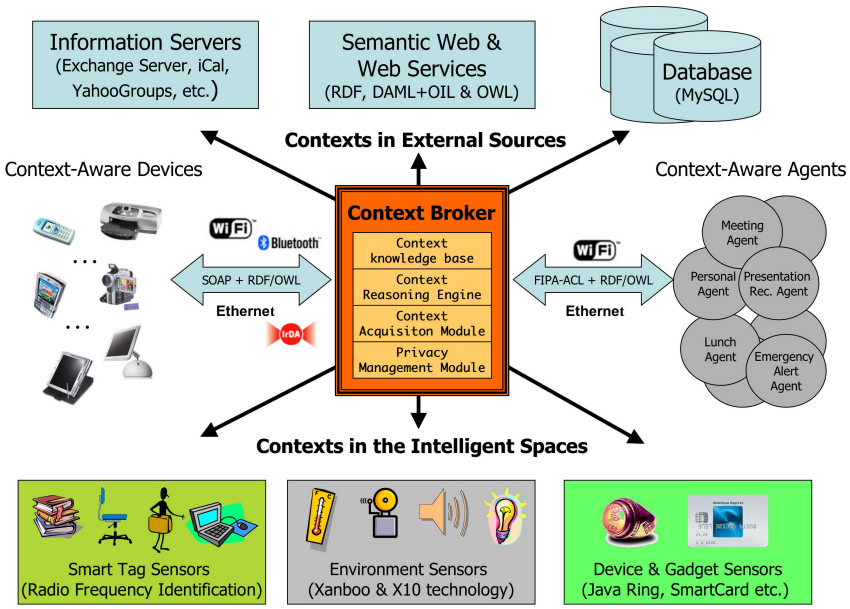

Abbildung 3.7: CoBrA Context Broker [CFJ04]

Einen anderen Ansatz wählen Hong und Landay und schlagen die **Context-aware Infrastructure** [HL01], eine infrastrukturbasierte Middleware, vor. Auf diese kann über ein Netzwerk zugegriffen werden. Im Unterschied zu den vorherigen Ansätzen kann über eine entsprechende Infrastruktur eine starke Entkopplung erreicht werden, wodurch eine größere Neutralität in Bezug auf die Hardware-Plattformen, die verwendeten Programmiersprachen und die Diversität der Applikationen, die auf eine solche Infrastruktur zugreifen können, entsteht. Auch eine Vereinfachung der angeschlossenen Komponenten kann erreicht werden, da die Infrastruktur verschiedene Funktionen übernehmen kann. Die Herausforderungen einer kontext-bewussten Infrastruktur bestehen in dem Design und der Entwicklung geeigneter Datenformate und Kommunikationsmechanismen, der Sicherheit und Privacy sowie der Skalierbarkeit der Infrastruktur.

In allen diesen Architekturen ist es das Ziel, die Erfassung und Nutzung von Kontext zu trennen. Trotz der verschiedenen Anpassungen und Modifikationen kann ein gemeinsames Architekturkonzept für kontext-bewusste Systeme identifiziert werden [BDR07]. Um die Funktionen eines kontext-bewussten Systems zu erfüllen, beinhaltet eine Architektur verschiedene Komponenten. Diese Komponenten eines

kontext-bewussten Systems umfassen die *Verarbeitungskomponente*, die *Verwaltungskomponente* und die *Verteilungskomponente*, die in der Regel als Middleware-Schicht des Systems betrachtet werden, sowie *Sensoren*, die die Informationen erfassen und *Applikationen*, die Dienste für einen Benutzer des kontext-bewussten Systems bereitstellen (siehe Abbildung 3.8).

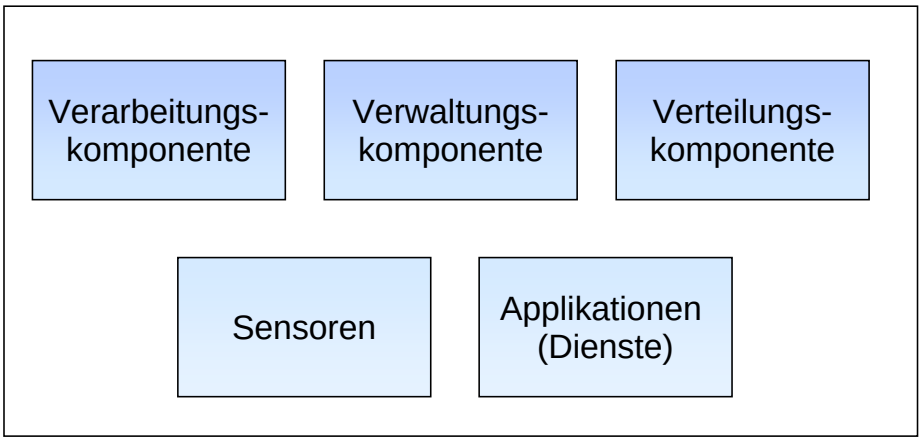

Abbildung 3.8: Komponenten eines kontext-bewussten Systems

Für die Erfassung der Kontextinformationen werden **Sensoren** eingesetzt. Die von den Sensoren erfassten Informationen werden für die weitere Verarbeitung vom System abgefragt. Dafür werden geeignete Treiber für Hardware-Sensoren und APIs für Software-Sensoren verwendet.

Die erfassten Kontextinformationen werden von der **Verarbeitungskomponente** aggregiert und für die Verwendung aufbereitet. Die Kontextinformationen können in dieser Komponente kombiniert oder interpretiert werden. Bei der Kombination werden verschiedene Sensordaten, die zu einem Bereich gehören, zusammengefasst und bei der Interpretation werden abgeleitet Informationen erzeugt. Die Kombination und Interpretation der Kontextinformationen kann jedoch auch durch den jeweiligen Dienst durchgeführt werden. Eine Interpretation der Kontextinformationen in der Verarbeitungskomponente des Systems erzeugt einen höheren Grad der Spezialisierung, als die Interpretation der Kontextinformationen durch den jeweiligen Dienst. Wird die Interpretation von der Verarbeitungskomponente vorgenommen, so erleichtert das die Entwicklung passender Dienste. Jedoch legt es die Architektur auf einen begrenzten Anwendungsbereich fest, zum Beispiel bei der Interpretation von Positionsinformationen als entsprechende Räume, die jeweils nur für ein bestimmtes Gebäude gültig sind. Die Interpretation beim Dienst erzeugt diese Beschränkungen nicht.

Die **Verwaltungkomponente** führt die verschiedenen Verwaltungsaufgaben des kontext-bewussten Systems durch. Diese beinhalten beispielsweise die Verwaltung der verfügbaren Komponenten, Management der benötigten Kommunikation zwischen den einzelnen Komponenten und die Speicherung von Kontextinformationen.

Um die Kontextinformationen in den von den Applikationen des kontext-bewussten Systems bereitgestellten Diensten nutzen zu können, müssen diese von der **Verteilungkomponente** zur Verfügung gestellt oder weitergegeben werden. Dafür muss ein Interface für die Applikationen bereitgestellt werden. Die Verteilung des Kontexts kann entweder synchron oder asynchron erfolgen. Synchrone Zugriffe auf die Architektur werden beispielsweise über RPCs (*Remote Procedure Calls*) realisiert. Dabei fragt eine Applikation bei Bedarf die Kontextinformationen ab. Asynchronen Zugriffe werden normalerweise über Subscriptions realisiert. Dafür spezifiziert eine Applikation zum Beispiel spezielle Events an denen sie interessiert ist.

Die **Applikationen** stellen Dienste bereit, welche auf den Kontext beziehungsweise auf spezielle Events reagieren und darauf hin dem Benutzer einen bestimmten Dienst bereitstellen. Die Applikationen und Services werden in manchen Ansätzen integriert, als Komponente des kontext-bewussten Systems dargestellt, in anderen Ansätzen als externe eigenständige Anwendungen verstanden.

Architekturen für kontext-bewusste Dienste sind fast immer auf spezielle Einsatzszenarien ausgerichtet, wie smarte Räume, und in der Regel auf einzelne bis wenige Dienste spezialisiert, zum Beispiel im Bereich Telekommunikation oder Touristenguides. Das erklärt auch, warum so viele unterschiedliche Ansätze existieren. Die Forschungsarbeiten auf diesem Gebiet werfen eine ganze Reihe von Forschungsfragen auf. Die Bereitstellung von einheitlichen Basisstrukturen oder generischen Frameworks für kontext-bewusste Dienste stellt immer noch eine Herausforderung dar. In vielen Forschungsarbeiten werden Middleware-Architekturen als geeignete Lösungen für die Bereitstellung kontext-bewusster Dienste vorgeschlagen. Beispiele dafür sind ein Middleware-Ansatz für kontext-bewusste Systeme in ubiquitären Umgebungen [dREdS08] oder eine Kontext-Middleware, die eine Entwicklung von adaptiven kontext-bewussten Applikationen für mobile Umgebungen ermöglicht [KPM05]. Eine Diskussion weiterer ähnlicher Ansätze findet sich jeweils auch in den genannten Arbeiten oder in verschiedenen Surveys wie [CK00] oder [BDR07].

3.2 Vernetzte virtuelle Umgebungen

Vernetzte virtuelle Umgebungen sind simulierte Umgebungen, die bestimmten Regeln und Prinzipien gehorchen und an denen mehrere Benutzer teilnehmen können. In der Literatur finden sich unterschiedliche Begriffe für die Bezeichnung solcher vernetzter virtueller Umgebungen. Beispiele sind "*networked virtual environment*" (vernetzte virtuelle Umgebung) [SZ99], "*distributed virtual environment*" (verteilte virtuelle Umgebung) [LLHL07], "*collaborative virtual environment*" (gemeinschaftliche virtuelle Umgebung) [BB04] oder auch "*Multiplayer*" oder "*Multi-User Virtual Environment*" [GLK+94].

Die Bezeichnung *Networked Virtual Environment* beschreibt die hier betrachteten vernetzten virtuellen Umgebungen am Besten, da sie die Eigenschaften der technischen und sozialen Vernetzung am Besten wiedergibt. Die Verwendung dieses Begriffs ist jedoch heutzutage selten geworden. Im Sprachgebrauch hat sich die Bezeichnung "virtuelle Welt" (*virtual world*) etabliert. Im weiteren Verlauf dieser Arbeit werden daher die Begriffe "virtuelle Welt" und "*Networked Virtual Environment*" synonym verwendet.

Genau genommen beinhaltet die Menge der virtuellen Welten neben den Vertretern, die auf einer technologischen Vernetzung basieren, auch solche virtuellen Umgebungen, die auf das System des Benutzers begrenzt sind (*offline*) und nur einen Teilnehmer erlauben (*single user / Singelplayer*). Jede virtuelle Welt bietet dem Benutzer eine virtuelle Umgebung in der der Benutzer, repräsentiert durch einen virtuellen Stellvertreter (virtuellen Charakter oder Avatar), an der virtuellen Welt teilnehmen und mit ihr interagieren kann. Besonders solche virtuellen Umgebungen, die einer großen Menge von Benutzern die Möglichkeiten bieten, Teil einer virtuellen Welt zu sein und gemeinschaftlich an Aufgaben zu arbeiten, bieten für wissenschaftliche Betrachtungen und insbesondere für diese Arbeit ein besonders interessantes Szenario, da dort eine Vielzahl unterschiedlicher Interaktionen und Kooperationen zwischen den Benutzern stattfinden.

3.2.1 Virtuelle Welten

Virtuelle Welten werden in dieser Arbeit verstanden als virtuelle Umgebungen, die im Internet gehostet oder in einem Netzwerk von einem Server bereitgestellt werden (*online*) und an denen mehrere Benutzer (*multiuser / Multiplayer*) zur selben Zeit teilnehmen und innerhalb der virtuellen Umgebung miteinander sowie mit der virtuellen Umgebung selbst interagieren können. Eines der Hauptmerkmale eines *Networked Virtual Environments* (NVE) ist das Vorhandensein eines Netzwerks, als Medium für den Austausch von Daten und Informationen zwischen den verschiedenen Teilnehmern eines gemeinsamen Erlebnisses [SZ99].

Eine gute Zusammenfassung der Thematik um NVEs findet sich bei Smed et. al. [SKH02b], die neben der technische Realisierung virtueller Welten auch allgemein die geschichtliche Entwicklung von verteilten interaktiven Echtzeit-Applikationen beschreiben. Sie unterscheiden drei Typen: Simmulationen,

Virtual Environments und Computerspiele. Abbildung 3.9 zeigt die Beziehungen der drei Applikationstypen.

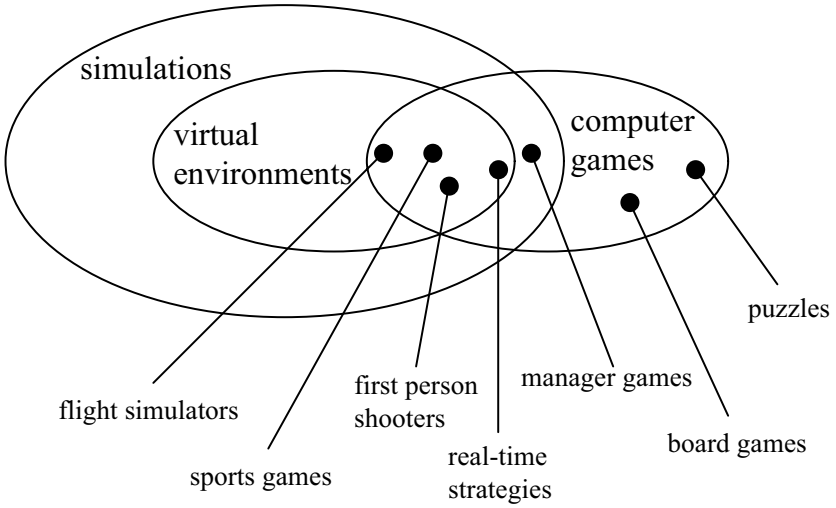

Abbildung 3.9: Beziehung zwischen Simulationen (*simulations*), virtuellen Umgebungen (*virtual environments*) und Computerspielen (*computer games*) [SKH02b]

Virtuelle Welten bieten ihren Benutzern simulierte Realitäten, die in der Regel physikalischen Gesetzmäßigkeiten wie Gravitation und anderen Prinzipien der realen Welt folgen, jedoch meist in einem Fantasy- oder Science Fiction- Szenario angesiedelt sind. Wichtige Eigenschaften dieser virtuellen Welten sind Repräsentationen der Benutzer innerhalb der virtuellen Welt, vielfältige Interaktionsmöglichkeiten und assoziierte Communities. Die Gruppe, die sich aus der gemeinsamen Schnittmenge bildet (siehe Abbildung 3.9), hat sich zum interessantesten und wichtigsten Vertreter der NVEs entwickelt und wird heute als "Multiplayer Online Games" bezeichnet.

3.2.2 Multiplayer Online Games als Vertreter virtueller Welten

Die wirtschaftlich erfolgreichsten und am weitesten verbreiteten virtuellen Welten findet man in *Multiplayer Online Games* (MOGs). Als "*Games*" werden in dieser Arbeit Computer- oder Konsolen-Spiele verstanden (auch bezeichnet als elektronische oder digitale Spiele) und als "*MOGs*" Spiele, die mit einem Computer oder einer Konsole über ein Netzwerk oder das Internet gespielt werden, mehr als einen Benutzer gleichzeitig in derselben Spielwelt unterstützen und die Interaktion in und mit der Spielwelt ermöglichen.

Eine einheitliche Definition des Wortes "Spiel" gibt es in der Literatur nicht, jedoch finden sich Sammlungen wichtiger Merkmale, die Spiele auszeichnen. Eine Zusammenfassung der Spielelemente aus verschiedenen Definitionen findet sich in [SZ04]. Eine sehr anschauliche und wegweisende Erklärung für das, was Spielen (*play*) ausmacht, stammt von Huizinga aus dem Jahre 1938 [Hui55]. Diese Erklärung bildet die Grundlage vieler aktueller Definitionen auch wenn einige Punkte heute überholt sind. Huizinga charakterisiert Spielen als:

- eine freie Aktivität,
- außerhalb des normalen Lebens,
- ohne ernsten Hintergrund (*not serious*),
- fesselnd,

- nicht verbunden mit materiellem Interesse oder Profit,
- findet innerhalb seiner eigenen Grenzen von Raum und Zeit statt,
- unterliegt einem geordneten Ablauf und festen Regeln,
- fördert die Bildung von sozialen Gruppen,
- unterstützt die Entwicklung von Gemeinschaften.

Online Games gibt es seit den 70er Jahren. Pioniere in diesem Bereich waren Richard Bartel und Roy Trubshaw, die 1978 das erste sogenannte *Multi-User-Dungeon* (MUD) an der Essex University entwickelten [Bar90]. Das MUD war textbasiert und das Spielkonzept basierte auf dem D&D (Dungeons & Dragons [78]) Rollenspielsystem. Heutzutage gibt es die verschiedensten Typen von MOGs (MOG Genre), wobei sich auch immer wieder neue Mischformen entwickeln. Eine grundlegende Kategorisierung von MOGs ist nicht ganz einfach und es finden sich auch immer wieder unterschiedliche Auslegungen. In seiner Taxonomie für digitale Spiele schlägt Jantke [Jan06] beispielsweise eine Unterscheidung der Spiele nach Typ, Genre (im engeren Sinne) und Klasse vor. Dabei bezeichnet der Typ des Spiels die verwendeten Spielmechanismen (logische und technische Basis wie Spielbrett, Würfel, aber auch Client-Server Architektur), das Genre die Art der Welt in der das Spiel stattfindet (Weltraum, Fantasiewelt, Epoche, ...) und die Klasse die Spielweise (Geschicklichkeit, Rätsel, ...).

Solche Kategorisierungen sind für die Beschreibung einzelner MOGs zwar gut geeignet, stellen aber nicht zwangsläufig Unterscheidungskriterien für verschiedene Genre dar. Um den Bereich MOG darstellen und auch abgrenzen zu können, ist somit eine detaillierte Betrachtung der verschiedenen Ausprägungen nötig. In den folgenden Beschreibungen von vier grundlegenden **Genres** von MOGs werden Eigenschaften der Spiele aus dem jeweiligen Bereich dargestellt:

- **Rollenspiele** (*Role-Playing Games* - RPGs) haben sich aus Pen&Paper-Rollenspielen entwickelt und haben als zentrales Element den Spielcharakter, der im Laufe der Zeit vom Spieler weiterentwickelt und geformt wird. Die Basis von Rollenspielen bilden Wertetabellen anhand derer alle Resultate und Aktionen ausgewürfelt werden. Typische Rollenspielewelten sind in einem Fantasyszenario angesiedelt aber auch Science Fiction oder andere Szenarien sind immer öfter Austragungsort dieser Spiele. Herausforderungen für den Spieler sind das Verständnis und die Kontrolle komplexer Systeme und Zusammenhänge.
- **Shooter** (*First Person Shooter* - FPS) haben zwei grundlegende Eigenschaften. Zum einen geht es um den Wettkampf mit unterschiedlichen Arten von Schusswaffen und zum anderen sieht der Spieler die Spielwelt durch die Augen seines Spielcharakters (im Gegensatz zu den RPGs bei denen der Spieler in der Perspektive hinter seiner Spielfigur steht). Für den Spieler liegt bei diesen Spielen die Herausforderung in der Hand-Auge-Koordination, da schnelle Reaktionszeiten und Genauigkeit entscheidend sind, wodurch diese Spielekategorie die erste war, die auch in sportlichen Tunieren ausgetragen wird (*e-sports*). Auch in dieser Kategorie gibt es Spiele, die in diversen Szenarien (Science Fiction, Fantasy, Weltkrieg, ...) angesiedelt sind.
- **Echtzeitstrategie** (*Real-Time Strategy* - RTS) Spiele stellen dem Spieler Optimierungsprobleme. Der Spieler muss mit knappen Ressourcen eine möglichst effektive Strategie verfolgen, wobei er eine Bandbreite von Möglichkeiten hat und gleichzeitig auf die Vorgehensweisen der anderen Parteien reagieren muss. Kennzeichnend sind für diese Spiele, dass eine Vielzahl an kleinen Spielcharakteren (Einheiten) zu kontrollieren sind anstelle von nur einem großen (wie der Spiel-Charakter bei FPS). Die Herausforderungen an den Spieler sind Stratgieentwicklung und Micro-Management. Auch hier gibt es diverse Spielwelt-Szenarien; beliebt sind zum Beispiel Science Fiction, historische Welten oder auch Fantasy.

- **Rennspiele** (*Racer*) haben als zentrales Element einen Rennwagen (Motorrad, Raumschiff, Flugzeug, etc.) mit dem es gilt, schneller oder auch kunstvoller eine vorgegebene Strecke zu absolvieren als die Konkurrenten, wobei hauptsächlich die Geschicklichkeit und das Timing eine Herausforderung an den Spieler darstellen. Die Spielwelt besteht normalerweise aus einer Rennstrecke, die jedoch wieder in verschiedene Szenarien eingebettet sein kann.

An den Beschreibungen der Spielekategorien kann man erkennen, dass eine Einordnung der Spiele nicht trivial ist. Unterscheidungsmerkmale, wie die Art der Spielwelt, sind bei der Beschreibung einzelner MOGs zwar hilfreich, sagen aber noch nichts über das vorliegende MOG Genre aus. Um die Menge der MOGs erfassen und kategorisieren zu können, müssen weitere Unterscheidungsmerkmale eingeführt werden, die eine solche Einordnung ermöglichen.

Bei einer in Rahmen dieser Arbeit durchgeführten Analyse von MOGs haben sich vor allem drei **Eigenschaften** als geeignet gezeigt, anhand derer eine Einordnung von MOGs vorgenommen werden kann. Diese Eigenschaften sind:

- **Persistenz der Spielwelt**: die Persistenz gibt an, wie lange dieselbe Spielwelt bestehen bleibt bevor sie zurückgesetzt / neu initialisiert wird. Übliche Größen sind dabei unter einer Stunde ($<h$), weniger als ein Jahr ($<Jahr$) oder länger als ein Jahr ($<unendlich$).

- **Entwicklungsanteil**: der Entwicklungsanteil gibt an, wieviel Einfluss der Spieler auf die virtuelle Umgebung oder den virtuellen Charakter nehmen kann. Sind alle Werte vorgegeben (*Vorgegebene Werte*), kann der Spieler einen einzelnen Charakter entwickeln (*Charakterentwicklung*) oder eine ganze Zivilisation (*Zivilisationsentwicklung*).

- **Perspektive (*View*)**: die Perspektive gibt an, welchen Blick der Spieler auf das Spiel hat. Schaut der Spieler mit den Augen seines virtuellen Characters in das Spiel (*First Person*), sieht er die Spielwelt als stünde er hinter seinem Charakter (*Third Person*) oder aus der Vogelperspektive (*Omnipresent*).

Auf Basis der Eigenschaften von MOGs können beispielsweise Spieleraktionen bezüglich des Interaktionsmodells und der Perspektive klassifiziert werden [CC06]. Neben den Spielen, die sehr genau einem Bereich zugeordnet werden können, haben sich auch diverse Mischformen entwickelt. Die Vielfalt der existierenden Spiele ist enorm und entwickelt sich stetig weiter, die in Abbildung 3.11 gewählten Kandidaten sind typische Exemplare, an Hand derer das Spektrum aber auch die Komplexität der Eigenschaften von MOGs sichtbar werden. Implizit können aus diesen grundsätzlichen Eigenschaften weitere Aussagen über die Spiele abgeleitet werden.

Bei der Unterscheidung von MOGs gibt es noch weitere Merkmale, nach denen die Spiele klassifiziert werden können, wie etwa die Spieleplattform (PC, Konsole oder Mobile Endgeräte). Die Übergänge zwischen verschiedenen Kategorien von Spielen können je nach Einteilung fließend sein, denn die Spielelemente, die typisch für ein Genre sind, können sich auch in Spielen anderer Genres wiederfinden. Einzelne Spiele lassen sich fast in beliebigem Maße differenzieren, was für die generelle Betrachtungen von MOGs jedoch keinen weiteren Mehrwert bedeutet.

Zur Unterscheidung von MOGs wird zusätzlich eine gemischt technologie-orientierte Betrachtung angewendet. Zum einen wird anhand der Persistenz der Spielwelt unterschieden und zum anderen anhand des Zugangs. Somit ergeben sich folgende MOG **Kategorien**:

- **MMOG** (Massively MOG): auch wenn der Name andeutet, dass es sich hierbei um Spiele handelt, die von sehr vielen Teilnehmen gleichzeitig und zusammen gespielt werden können, so ist dies doch eigentlich nur ein Aspekt, den diese Spiele haben sollen (vor allem die technische Realisierung solcher Systeme ist heute nur eingeschränkt möglich und wird durch verschiedene Kompromisslösungen umgesetzt). Das eigentliche Merkmal dieser Kategorie ist die Persistenz der Spielwelt. Es handelt sich bei MMOGs um Client-Server Systeme, wobei die Server-Seite die Spielwelt beherbergt, die kontinuierlich weiter läuft (auch wenn sich kein Teilnehmer in der Welt befindet).

MMOGs haben kein erreichbares Ende. Sie setzen sich deshalb endlos fort und "enden" theoretisch erst wenn das Spiel eingestellt wird. Für MMOGs gibt es Vertreter aus verschiedenen Bereichen. Treiber dieses Bereichs und am weitesten verbreitet sind aber die MMORPGs (Massively Multiplayer Online RolePlaying Games).

- **SMOG** (Session MOG): Session-MOGs werden meist einfach nur als MOGs bezeichnet. Ein (S)MOG ist in der Regel wie ein MMOG als Client-Server System realisiert. Seine Spielwelt wird jedoch für jede Runde oder Partie neu instantiiert, sodass multiple Kopien derselben Welt gleichzeitig existieren, die von unterschiedlichen Teilnehmern bevölkert werden. Diese Welten sind zeitlich und / oder durch andere Bedingungen begrenzt und bestehen in der Regel je nach Spiel zwischen wenigen Minuten bis hin zu mehreren Stunden. Typische Vertreter von (S)MOGs sind klassische Multiplayer FPS und RTS Spiele.
- **Browsergames**: Browsergames haben im Unterschied zu den anderen Kategorien keinen Client, der auf dem System des Spielers läuft. Wie der Name hier richtig andeutet, handelt es sich um Spiele, deren Zugang über einen Internetbrowser stattfindet. Browsergames gibt es beispielsweise aus den Bereichen RTS oder RPG.

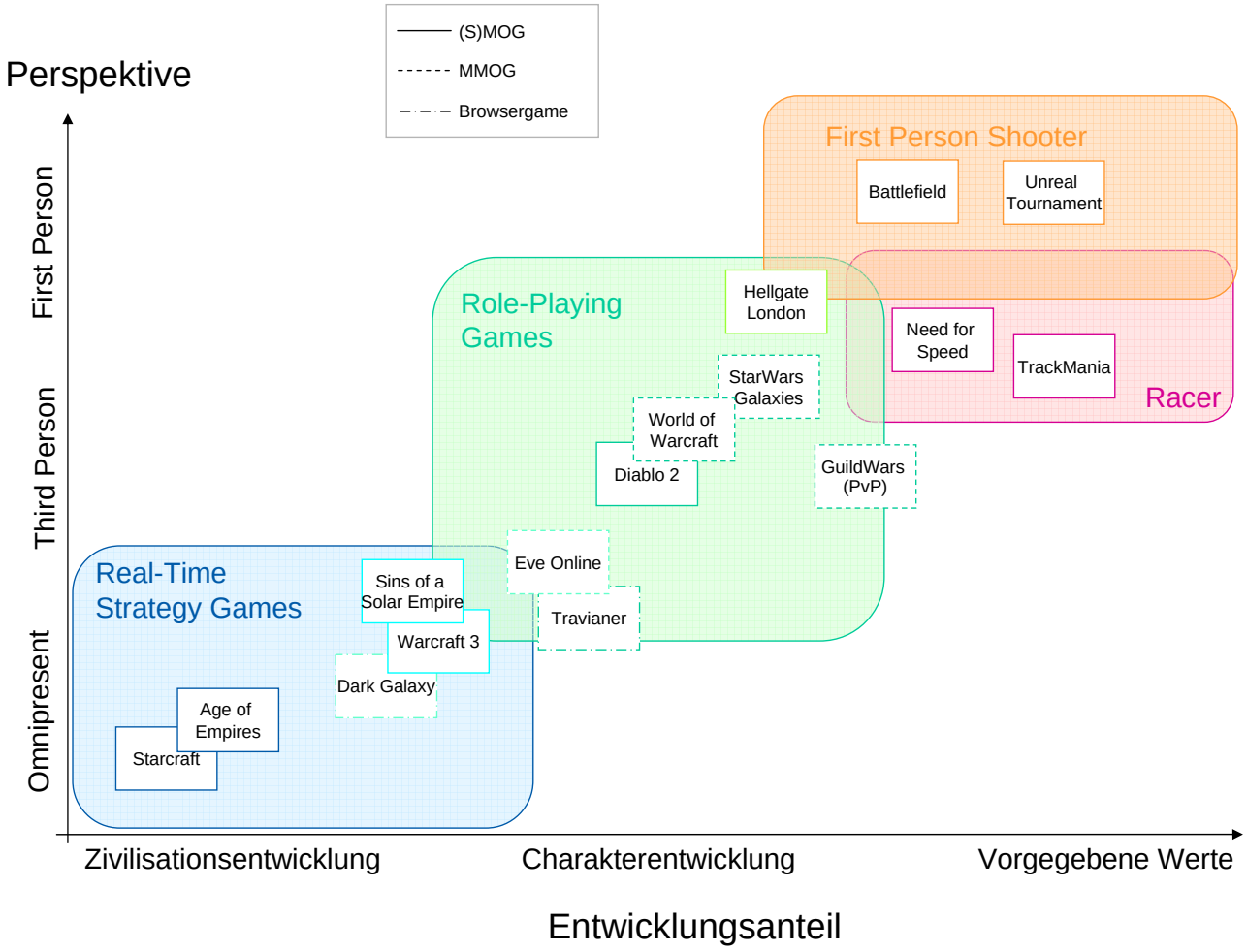

Abbildung 3.10: Beziehung zwischen Perspektive und Entwicklungsanteil bei MOGs

Bei der Analyse der MOGs hat sich gezeigt, dass Entwicklungsanteil und Perspektive in Beziehung zueinander stehen (siehe Abbildung 3.10). Es ergeben sich drei Cluster von Spielen: (1) Spiele, die

primär aus der Vogelperspektive gespielt werden und dem Spieler die Entwicklung einer Zivilisation ermöglichen. Diese Spiele sind zumeist RTS-Spiele. (2) Spiele, die hauptsächlich aus der Third-Person-Perspektive gespielt werden und dem Spieler die Entwicklung seines Charakters ermöglichen. Diese Spiele kommen in der Regel aus dem Bereich der RPGs. (3) Spiele, die in erster Linie aus der First-Person-Perspektive gespielt werden und mit vorgegebenen Werten arbeiten. Dabei handelt es sich oft um FPS-Spiele. Es können jedoch auch Racer sein, wobei es dort oft die Möglichkeit gibt frei zwischen der First-Person- und der Third-Person-Perspektive zu wechseln.

Um MOGs hinsichtlich ihrer drei Basiseigenschaften einzuordnen kann man die beiden Eigenschaften (Entwicklungsanteil und Perspektive) zusammenfassen und somit auf einer Achse abbilden. Dadurch ergibt sich ein 2-Dimensionales Feld mit den Dimensionen Entwicklungsanteil / Perspektive und Persistenz (siehe Abbildung 3.11). Die Abbildung stellt zusätzlich den Unterschied der verschiedenen MOGs hinsichtlich ihrer Dauer dar. Die Sessions der (S)MOGs sind im Vergleich am kürzesten. Sie können nur wenige Sekunden dauern und in der Regel maximal mehrere Stunden. Am längsten dauern die MMOGs, die normalerweise nur durch das Abschalten der Spiel-Server beendet werden. Die ausgewählten Browsergames liegen hinsichtlich ihrer zeitlichen Dauer zwischen den (S)MOGs und den MMOG, da es typisch für Browsergames ist, dass eine "Runde" (die Zeit in der das Spiel läuft ohne neu gestartet zu werden) mehrere Monate dauert. Das ist jedoch kein hartes Kriterium. Es gibt auch Browsergames die in kurzen Sessions gespielt werden oder immer weiterlaufen.

Wie in den Abbildungen 3.10 und 3.11 angedeutet wird, sind die Übergänge zwischen den verschiedenen Ausprägungen von MOGs jedoch fließend. Es gibt Spiele, die nicht eindeutig einem Genre zugeordnet werden können, da sie bestimmte Spielelemente, wie den Entwicklungsanteil nur teilweise oder in einer Mischform einsetzen oder einen Wechsel der Perspektive, manuell oder je nach Spielsituation, beinhalten. Das Spiel *Warcraft III* [10] ist eigentlich ein RTS im klassischen Sinne und basiert auf der Entwicklung einer Zivilisation, jedoch beinhaltet es zusätzlich einen ausgezeichneten Spielcharakter, der wie bei einem RPG weiterentwickelt werden kann. Im RPG Hellgate London [27] ist die Verwendung verschiedener Perspektiven möglich. Es beinhaltet Spielcharaktere (wie den Schwertmeister), die, wie in anderen RPGs, aus der Third-Person-Perspektive gespielt werden, als auch Charaktere (wie den Scharfschützen) die in First-Person-Perspektive gespielt werden, wobei auch bei diesen Charakteren die Verwendung der Third-Person-Perspektive möglich ist. Auch hinsichtlich der zeitlichen Dauer gibt es Ausnahmen. So enthält das Spiel *GuildWars* [5], welches eigentlich aus der Kategorie der MMOGs kommt, beispielsweise innerhalb des Spiels Bereiche, die wie ein (S)MOG funktionieren, da sie als begrenzte Session mit bestimmten Teilnehmern realisiert sind und vom Spielprinzip eher wie ein (*Team-Deathmatch* oder *Capture the Flag*) gestaltet sind.

Im Allgemeinen ermöglicht das Benutzerinterface einer virtuellen Welt dem Benutzer, Aktionen in der virtuellen Welt auszuführen. Zum Beispiel ist der *virtuelle Charakter* (oder Avatar) in der Regel das wichtigste Element des Benutzerinterfaces. Mittels seines virtuellen Charakters kann der Benutzer verschiedenste Interaktionen mit der in-game Umgebung und deren Bevölkerung (*non player characters* - NPCs) durchführen, sowie mit den virtuellen Charakteren der anderen Benutzer interagieren. Der virtuelle Charakter ist intuitiv zu bedienen und ermöglicht dadurch einen leichten Einstieg für neue Benutzer. Oft kann ein Benutzer seinen virtuellen Charakter nach seinem Geschmack konfigurieren. Schon durch die Erstellung seines virtuellen Charakters kann der Benutzer also einen Teil von sich in die virtuelle Welt einbringen. Die größten Einflussmöglichkeiten bieten virtuelle Welten aus dem Rollenspiel-Bereich, Shooter bieten dagegen oft kaum Einflussmöglichkeiten und in Strategiespielen handelt es sich immer um eine ganze Charaktergruppe. Dabei hat der Benutzer in der Regel die Auswahl zwischen verschiedenen Rassen und Klassen und kann dann weitere Personalisierungen vornehmen, zum Beispiel durch die Vergabe eines Charakternamens oder die Konfiguration von Statuswerten oder Aussehen des Charakters. Durch das Betreten der virtuellen Welt mit seinem Charakter entsteht eine besondere Atmosphäre für den Benutzer, da nun sein "Abbild" in die virtuelle Umgebung eingebettet wird und somit ein immersives Szenario entsteht. Eine typische Erweiterung der Interaktionsmöglichkeiten zwischen verschiedenen Benutzern einer virtuellen Welt bieten integrierte Text-Chat Systeme. Ein Text-Chat ermöglicht meist die

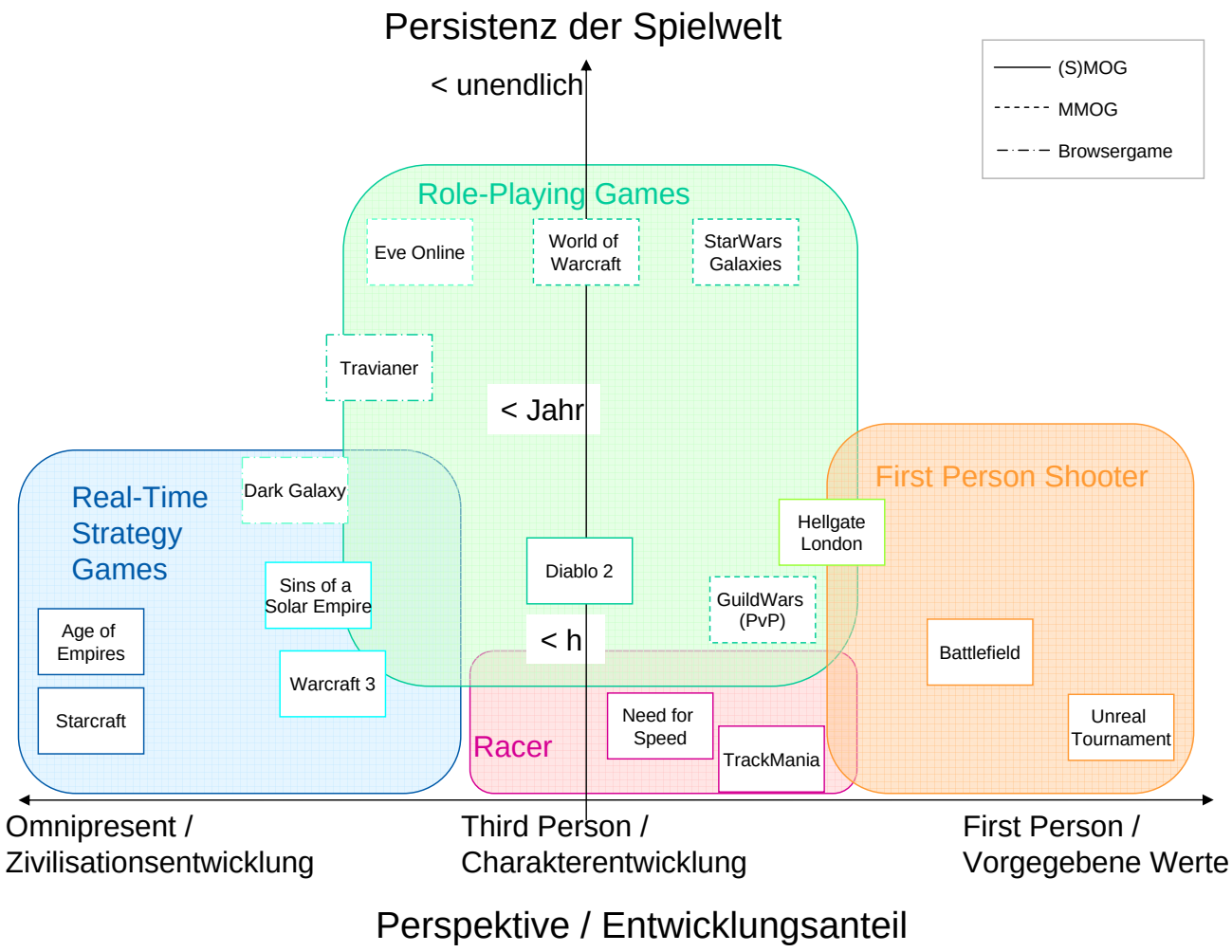

Abbildung 3.11: Einordnung von MOGs hinsichtlich der drei Basiseigenschaften

Verwendung unterschiedlicher Chat-Kanäle und verschiedene Kommunikationsmöglichkeiten, wie "private" Kommunikation zwischen zwei Benutzern, Gruppenkommunikation oder auch Chat-Kanäle, die alle Benutzer gemeinsam nutzen können (z.B. für den Handel).

Bedeutung von MMOGs

MMOGs haben rasant an Bedeutung gewonnen und stellen einen stark wachsenden Markt dar. Unter den MMOGs haben die *Massively Multiplayer Online Role-Playing Games* (MMORPGs) einen besonderen Stellenwert. Ihre Bedeutung für die Entwicklung der MMOGs und deren Markt ist gewaltig. Auch für wissenschaftliche Untersuchungen haben sie einen besonderen Stellenwert, da sie sowohl technische Fragestellungen aufwerfen als auch komplexe soziale Systeme darstellen [Yee06]. MMORPGs haben in den letzten 10 Jahren stark steigende Nutzerzahlen (siehe Subscription Statistiken [79] [68]), Verbreitung und wirtschaftliche Bedeutung zu verzeichnen und wachsen stetig weiter. Der Boom der MMORPGs wurde in erster Linie durch das Spiel *World of Warcaft* (WoW) [11] von Blizzard Entertainment ausgelöst, das inzwischen alleine über 11,5 Millionen Teilnehmer hat [12]. Dabei ist *World of Warcaft* in verschiedener Hinsicht maßgeblich, zum Beispiel hinsichtlich der Benutzerzahlen (siehe auch Abbildung 3.12), der weltweiten Verbreitung (Europa, Amerika und Asien), der Eröffnung neuer Zielgruppen (Massentauglichkeit), dem Spieldesign (das heute eine Referenz darstellt und von vielen anderen Spielen kopiert

wird) oder der Öffnung gegenüber der Community (durch Spielmodifikationen und Charakterdaten-Präsentation). Darüber hinaus zeichnet es sich durch einen besonderen Stellenwert des Teamplays, der Kommunikation und Kooperation aus, wodurch es in dieser Arbeit an verschiedenen Stellen auch als Referenz verwendet wird.

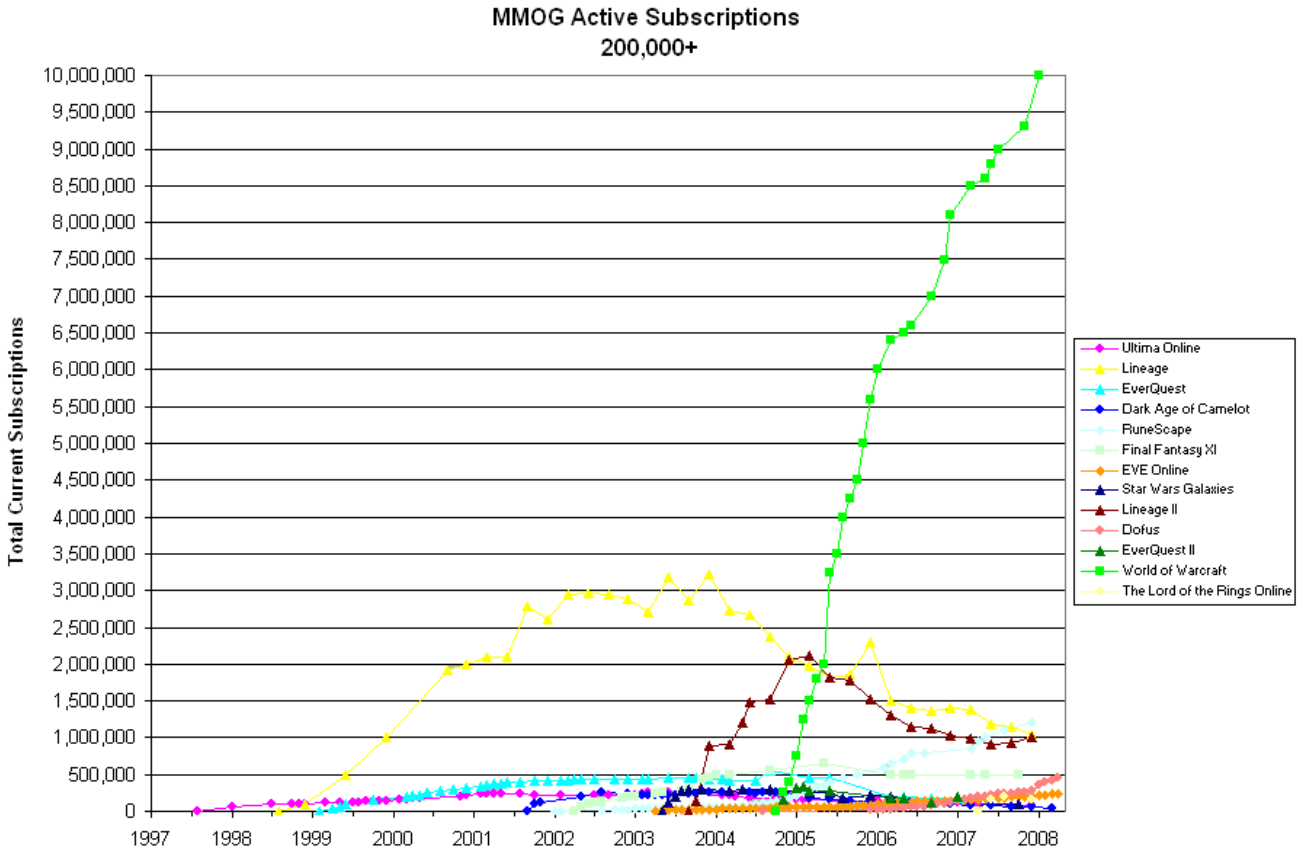

Abbildung 3.12: MMOG Subscriptions für MMOGs mit mindestens 200.000 Teilnehmnern (MMOGCHART.COM [79])

3.2.3 Technische Grundlagen von MOGs

MOGs sind verteilte Echtzeitsysteme, die besondere Anforderungen bezüglich des Kommunikationsnetzwerkes, der Antwortzeiten, der Skalierbarkeit und auch der Sicherheit stellen. Netzwerkverkehr, den Spiele erzeugen (*Game traffic*), hat eine besondere Form. Beispielsweise ist er in der Regel eher unregelmäßig (*bursty traffic*) und besteht aus sehr vielen kleinen Paketen. Verschiedene Analysen und Untersuchungen haben gezeigt, wie sich *Game traffic* darstellt, siehe [FCFW05], [CFSS05], [CHL06a], [KCC+05] oder [FÖ02]. Darüber hinaus wird in verwandten Arbeiten untersucht, welche Auswirkungen dies auf die verwendeten Kommunikationsmechanismen hat ([HPGH07], [WCC+09]) und welche Mechanismen in MOGs eingesetzt werden, vor allem um den entstehenden Verzögerungen (*Latenzen*) zu begegnen ([Ber01], [BKV06]).

Die Eignung verschiedener Architekturen für MOGs, sowie neue Ansätze zur Realisierung werden in der Forschung diskutiert. Dies wird im Folgenden kurz vorgestellt. Die betrachteten Architekturen beinhalten verschiedene Typen von Client-Server Systemen sowie *Peer-to-Peer* (P2P) [SE05] basierte Ansätze.

Architektur

Die heute übliche Architektur zur Realisierung von MOGs ist Client-Server (siehe Abbildung 3.13). Auf dem Server gesammelte Events werden dabei an die Spieler verteilt [BVHG09]. Dabei gibt es unterschiedliche Realisierungsformen, wie die Verwendung von verschiedenen Servern für verschiedene Kopien derselben virtuellen Welt oder die Verwendung von Server-Clustern, die die virtuelle Welt in Regionen oder Zonen unterteilen, sie jedoch für den Benutzer als eine konsistente Welt darstellen. Die Kommunikation zwischen dem Spiel-Client und dem Spiel-Server sollte so effizient wie möglich umgesetzt werden, denn hohe Latenzen haben nachweislich negative Auswirkungen auf das Spielen und die Performance der Spieler. Dabei variieren die Anforderungen an die Spieldatenübertragung je nach Spiel basierend auf den Aktionen, die im Spiel ausgeführt werden [CC06]. Verschiedene Untersuchungen haben gezeigt, dass aufgrund zu hoher Latenzen nicht nur die Spieler schlechter werden, sondern auch die Qualität der Interaktion leidet [DWW05] und Spieler schneller das Spiel wieder verlassen [CHL09].

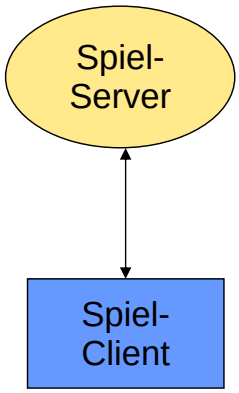

Abbildung 3.13: MOG Client-Server Architektur

Client-Server basierte Systeme sind weit verbreitet, da sie verschiedene Vorteile gegenüber verteilten Architekturen bieten, dazu zählen ein einfacheres Management, Konsistenz des Zustandes, einfachere Umsetzung von Sicherheitsmechanismen und die Anonymität der Clients untereinander. Die größten Nachteile zentraler System sind Skalierbarkeitsprobleme (Server ist der Flaschenhals *bottleneck* der Architektur), und dass es einen *Single Point of Failure* (einen Punkt dessen Ausfall den Ausfall des kompletten Systems verursacht) gibt. In P2P-Architekturen werden die Zustandsinformationen eines Spieles (*game state*) verteilt auf den einzelnen Peers der Spieler verwaltet. Es gibt keinen zentralen Server. Die großen Herausforderungen bei einer P2P-Architektur sind, den Spielzustand konsistent zu halten und Zuverlässigkeit als auch Sicherheit zu gewährleisten. Betrachtungen und Vergleich von Client-Server und P2P-Systemen finden sich unter anderem bei [MFW02], [VGH06], [SE05] oder [SKH02a].

In der Literatur werden Design-Alternativen vorgestellt und diskutiert, vor allem um die Skalierbarkeit von MOGs zu verbessern. P2P-Ansätze sind dabei im Allgemeinen sehr erfolgreich, werden im Spielebereich jedoch hauptsächlich in der Forschung verwendet. Dabei gibt es verschiedene Ansätze P2P-basierter Spielearchitekturen (wie [HL04] oder [YMYI05]) und verteilter virtueller Umgebungen (wie [DMLS04] oder [LLHL07]). In kommerziellen Produkten werden im allgemeinen keine P2P-Architekturen verwendet, da die Sicherheit in P2P-Systemen nicht ausreichend gewährleistet werden kann. Die Vorteile der einfacheren Verwaltung einer zentralisierten Architektur überwiegen [MFW02].

Untersuchungen, wie eine Lastverteilung ohne Verlust der zentralen Kontrolle realisiert werden könnte, werden in der Forschung durchgeführt und es werden verschiedene Ansätze für die Verbesserung der Skalierbarkeit von Spielen vorgeschlagen. Am meisten Relevanz haben dabei hybride Ansätze, die versuchen, die Skalierbarkeitsprobleme zu überwinden. Architekturen, die auf der Verwendung von Proxy-Servern beruhen, zeigen sich für die Verwendung bei MMOGs als besser passend [VGH06]. Sie bilden einen Kompromiss, da sie aus einem zentralen Server und einer Menge an verteilten Proxy-Servern be-

stehen. Diese Proxies können hierarchisch angeordnet sein oder als Proxy-Pool. Auch hier gibt es in der Forschung verschiedene Ansätze ([Vik05], [BRS02], [MFW02]).

Neben der Verwendung von Replizierung kann eine Verbesserung der Skalierbarkeit des Client-Server-Ansatzes auch durch die Verwendung von verteilten Client-Server-Systemen (*distributed Client-Server*) erreicht werden. In diesem Fall werden die Aufgaben unter den verwendeten Servern aufgeteilt. Diese Aufteilung kann nach räumlichen, zeitlichen oder funktionalen Gesichtspunkten durchgeführt werden [CXTL02]. Um die entstehende Last aufzuteilen, wird die virtuelle Welt partitioniert, wobei jeder Server eine Partition zugeteilt bekommt. Da ein Spieler in einer virtuellen Umgebung immer nur einen gewissen Bereich wahrnehmen kann (*Area of Interesst* oder "Aura"), hat er auch nur Interesse an Informationen, die diesem Bereich entsprechen (*Interest Management*) [MBD00]. So können verschiedene Gebiete einer virtuellen Welt auf verschiedene Server aufgeteilt werden [AT06]. Diese Aufteilung liefert einen guten Kompromiss, da im Vergleich zu einer zentralisierten Client-Server-Architektur eine bessere Skalierung erreicht wird und im Vergleich zu einer verteilten P2P-Architektur Mittel bereitgestellt werden, um Cheating zu erkennen und zu verhindern. Die verteilte Client-Server-Architektur hat sich als weitestgehend geeignet gezeigt und wird auch in kommerziellen Spielen verwendet.

Bedeutung der Latenz

MOGs sind verteilte Anwendungen und somit werden sie durch verschiedene Aspekte der Netzwerkkommunikation beeinflusst [SZ99]. Besonders hohe Latenzen wirken sich negativ auf die Interaktion des Benutzers mit der virtuellen Umgebung aus.

Im Allgemeinen wird als Richtwert für interaktive Echtzeitsysteme eine Latenz von 0,1 bis 1,0 Sekunden als akzeptabel angegeben [SKH02a]. Untersuchungen von verschiedenen MOGs haben gezeigt, dass die Toleranz der akzeptierten Latenz je nach Spielgenre und den in den verschiedenen Genres typischen Aktionen variiert. Besonders Genres in denen der Spieler jede Aktion seines Charakters direkt steuert haben eine sehr geringe Verzögerungstoleranz. Beispiele dafür sind FPS Spiele [BCL+04] und Racer [PW02a]. Verschiedene Untersuchungen haben ergeben, dass bis zu einer Latenz von 100ms kaum Beeinträchtigungen auftreten, bis zu 150ms-180ms akzeptable Beeinträchtigungen auftreten und ab 200ms massive Beeinträchtigungen auftreten. Etwas toleranter sind Sportspiele [NC04] und MMOGs, deren akzeptabler Bereich bei bis zu 500ms-750ms liegt. Am wenigsten beeinträchtigt von einer höheren Latenz werden RTS Spiele [Cla05], die bis zu 2000ms ein akzeptables Resultat ermöglichen.

Gute Spieler sind sensibler gegenüber steigenden Latenzen und ihre Performance wird auch stärker beeinträchtigt als die nicht so guter Spieler [ZA04]. Dabei werden nicht nur die wahrgenommene Qualität und Performance der Spieler beeinträchtigt. Untersuchungen haben gezeigt, dass sich die Effekte der Netzwerkqualität auch auf die Verweildauer im Spiel auswirken [CHL06b]. Die Auswirkungen der Latenz auf ein Spiel, sind abhängig von den im Spiel eingesetzten Mechanismen Latenzen zu verbergen (wie *client-side prediction*) [Ber01]. So kann dadurch ein Spiel trotz hoher Latenzen akzeptable Bedingungen für den Spieler bieten (beispielsweise mehr als doppelt so viel Latenz tolerieren bei einem MMORPG wie *Everquest2* [FRS05]).

3.3 Online-Communities

Online-Communities sind eines der interessantesten Phänomene, die sich im Zusammenhang mit dem Internet entwickelt haben. Für Online-Communities wird in verwandten Arbeiten auch der Begriff *virtuelle Community* verwendet, dieser ist jedoch irreführend da es sich immer um reale Communities handelt. Als das Internet noch in seiner Anfangsphase steckte, definierte der Sozialwissenschaftler Rheingold [Rhe93] eine *Online-Community* als einen sozialen Zusammenschluss, der auftritt, wenn eine öffentliche Diskussion so lange und intensiv geführt wird, dass sich daraus Netzwerkstrukturen im Cyberspace bilden:

"Virtual communities are social aggregations that emerge from the Net when enough people carry on those public discussion long enough, with sufficient human feeling, to form webs of personal relationships in cyber-space."

Im Anschluss daran entstanden verschiedene Definitionen von Online-Communities. Die in Tabelle 3.1 gezeigte Übersicht bedeutender Definitionen wurde von Panten zusammengetragen [Pan05]. Panten zitiert unter anderem die Online-Community-Definitionen von Rheingold, Hagel und Armstrong und Preece (siehe Tabelle 3.1). Dabei betrachtet er insbesondere die in den Definitionen enthaltenen Dimensionen, wie sozialer Austausch, Interaktion, Zugang oder technische Plattform. Die grundlegenden Eigenschaften von Online-Communities finden sich in der von Panten zitierten Definition von Hagel und Armstrong:

"Virtual communities are [...] groups of people with common interests and needs who come together online."

Demnach zeichnen sich Online-Communities dadurch aus, dass sich eine Gruppe aus einem gemeinsamen Interesse heraus bildet und sich online drüber austauscht. Eine detailliertere Übersicht über die Eigenschaften von Online-Communities gibt Preece in seiner von Panten zitierten Definition. Laut Preece besteht eine Online-Community aus:

- **Personen**, die sozial interagieren im Streben danach ihre eigenen Bedürfnisse zu befriedigen oder bestimmte Rollen wie Leitung oder Moderation zu übernehmen,
- Einem **gemeinsamen Grund**, Interesse, Bedarf, Informationsaustausch, oder Dienst der eine Begründung für die Community darstellt,
- **Grundsätzen**, in der Form von stillen Voraussetzungen, Ritualen, Protokollen, Regeln und Gesetzten, die die Interaktionen zwischen den Personen leiten,
- **Computersystemen** für den Support und die Vermittlung sozialer Interaktionen und eines Zusammengehörigkeitsgefühls.

Hsu und Lu [HL07] fassen Online-Communities zusammen als Gruppen von Menschen, die über das Medium Internet miteinander kommunizieren, Ideen, Meinungen oder sonstige Inhalte austauschen und dabei unabhängig von ihrer geographischen Lage oder ethnischen Abstammung interagieren.

Die Basis einer Online-Community ist die Kommunikation und Interaktion zwischen den Teilnehmern. Die Entwicklung einer Community ist abhängig von den Beziehungen zwischen den Mitgliedern der virtuellen Gemeinschaft. Die schnelle Verbreitung und stetige Entwicklung des Internets haben den Weg für den Erfolg von Online-Communities bereitet. Besonders die Demokratisierung des Internets (Web 2.0) und die Vereinfachung der aktiven Partizipation (*user created content*) stellen die Basis für eine florierende Community-Kultur dar.

3.3.1 Ausprägungen von Online-Communities

Online-Communities lassen sich in unterschiedliche Klassen unterteilen. Eine umfassende Betrachtung verschiedener Kategorisierungen hat Äkkinen zusammengefasst [Äkk05]. Er unterscheidet folgende Klassifikationsarten: inhaltsbasiert und ertragsbasiert. Bei inhaltsbasierten Kategorisierungen wird auf Basis der Community zugrunde liegenden Motivation (beispielsweise *Communities of Practice*, *Online Learning* oder *Games*) sowie bezüglich der Initiatoren der Community unterschieden, d.h. entweder sind die Mitglieder selbst die treibenden Kraft hinter der Community oder es handelt sich um eine durch eine Organisation initiierte Community. Die ertragsbasierten Kategorisierungen beziehen sich auf die Rentabilität einer Community und finden eher Anwendung in einer Geschäftsmodell-Betrachtung. Laut

Tabelle 3.1: Ausgewählte Definitionen des Begriffs "Virtuelle Community" [Pan05]

Verfasser (Jahr)	Definition	Enthaltene Dimensionen
Rheingold (1993)	„Virtual communities are social aggregations that emerge from the Net when enough people carry on those public discussion long enough, with sufficient human feeling, to form webs of personal relationship in cyberspace.“	• Sozialer Austausch • Persönliche, über das Web vermittelte Beziehungen
Hagel III/ Armstrong (1997)	Virtual Communities are „[…] groups of people with common interests and needs who come together on-line. Most are drawn by the opportunity to share a sense of community with like-minded strangers, regardless of where they live. But virtual communities are more than just a social phenomenon. What starts off as a group drawn together by common interests ends up as a group with a critical mass of purchasing power, partly thanks to the fact that communities allow members to exchange information on such things as a product's price and quality. “ (S.147)	• Spezifischer Interessenschwerpunkt • Integration von Inhalt und Kommunikation • Von Mitgliedern bereitgestellte Informationen • Zugang zu konkurrierenden Anbietern • Kommerzielle Orientierung
Figallo (1998)	„In a true Community on the web, the member feels part of a larger social whole. There's an interwoven web of relationships between members. There's an ongoing exchange between members of commonly valued things. Relationships between members last through time, creating shared histories. “ (S. 15)	• Soziale Identität • Sozialer Austausch zu allgemein geschätzten Dingen • Stabile Beziehungen
Lechner et al. (1998)	„Virtuelle Gemeinschaften werden demnach betrachtet als eine organisierte Sammlung von Agenten auf der Basis einer durch die Informations- und Kommunikationstechnologie zur Verfügung gestellten Plattform. Die Mitglieder der Gemeinschaft werden durch Avatare auf der Plattform repräsentiert und die Organisation der Gemeinschaft wird auf der Plattform implementiert.“ (S. 3)	• Technische Plattform • Interaktion von (Software-)Agenten
Kozinets (1999)	„They (virtual communities of consumption) can be defined as affiliative groups whose online interactions are based upon shared enthusiasm for, and knowledge of, a specific consumption activity or related group of activities. “ (S. 254)	• Gemeinsames Konsuminteresse • Online-Aktivitäten
Schubert (1999)	„Virtuelle Gemeinschaften beschreiben den Zusammenschluss von Individuen oder Organisationen, die gemeinsame Werte und Interessen miteinander teilen und die über längere Zeit mittels elektronischer Medien, die orts- und (teilweise auch) zeitungebunden in einem gemeinsamen semantischen Raum (gemeinsame Begriffswelt) kommunizieren. “ (S. 30)	• Art der Personen • Gemeinsame Werte und Interessen • Elektronische Medien als Mittler der Kommunikation
Preece (2000)	„An online community consists of: • People, who interact socially as they strive to satisfy their own needs or perform special roles, such as leading or moderating. • A shared purpose, such as an interest, need, information exchange, or service that provides a reason fort he community. • Policies, in the form of tacit assumptions, rituals, protocols, rules, and laws that guide people's interactions. • Computer systems, to support and mediate social interaction and facilitate a sense of togetherness. “ (S 10)	• Soziale Interaktion • Gemeinsamer Zweck • Definierte Regeln der sozialen Interaktion • Technische Plattform

Äkkinen stammt die umfassenste Kategorisierung von Online-Communities (*Virtual Communities*) von Stanoevska-Slabeva und Schmid. Stanoevska-Slabeva und Schmid unterscheiden Online-Communities bezüglich der Bedürfnisse und Beweggründe ihrer Mitglieder (inhaltsbasierte Betrachtung). Sie definieren die Kategorien "Diskussion", "Aufgaben / Ziel orientiert", "Virtuelle Welt" und "Hybrid" (siehe Abbildung 3.14).

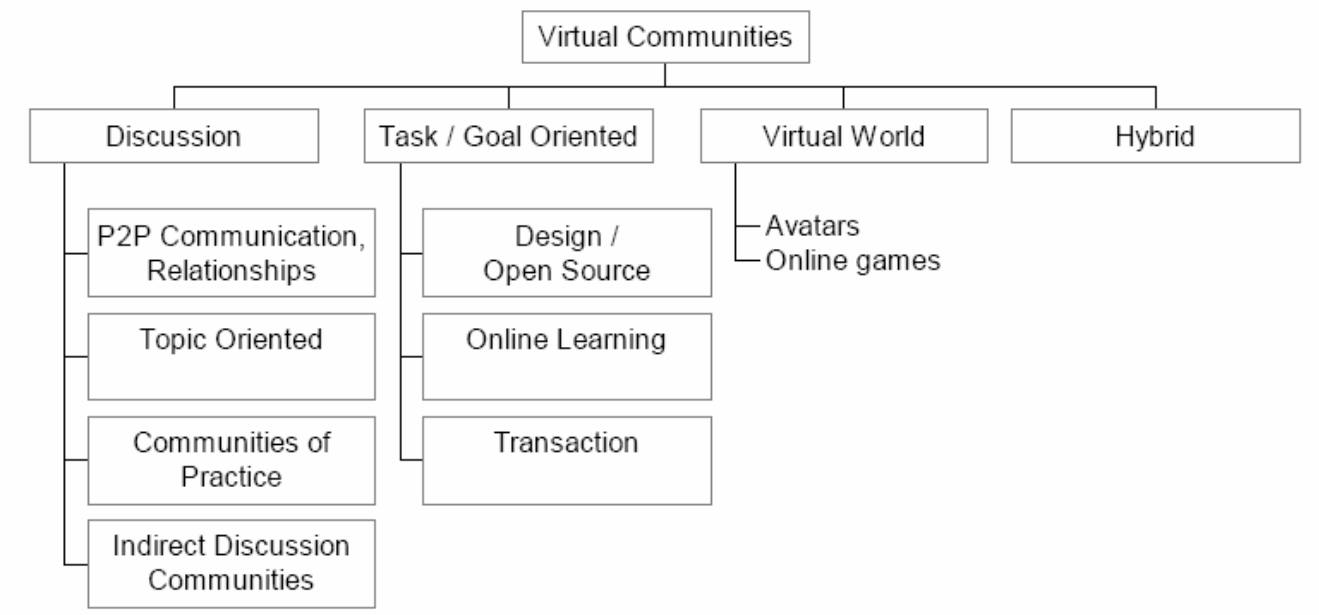

Abbildung 3.14: Klassifizierung virtueller Communities nach Stanoevska-Slabeva und Schmid [Äkk05]

- **Diskussions-Communities** (*Discussion*): bei Diskussions-Communities geht es primär um die Generierung und den Austausch von Inhalten, die einem bestimmten Thema zugeordnet sind. Es gibt vier Untergruppen von Diskussions-Communities: person-to-person Communities (*P2P Communication, Relationships*) wie soziale Gemeinschaften, themenbezogene Communities (*Topic Oriented*) wie Sportgemeinschaften, *Communities of Practice* zum Beispiel in der Software Entwicklung und indirekte Diskussions-Communities (*Indirect Discussion Communities*) wie die Amazon Review-Community.
- **Aufgaben- / Zielorientierte Communities** (*Task / Goal Oriented*): Gemeinschaften, die sich durch die Zusammenarbeit beim Streben nach einem gemeinsam Ziel auszeichnen. Auch hier gibt es drei Untergruppen: *Design-Communities* (z.B. *Open Source* Software-Entwicklung), *Online Learning* und *Transaction-Communities* aus dem Bereich e-Commerce.
- **Virtuelle Welt** (*Virtual World*): Online-Communities, die rund um virtuelle Welten und Spiele entstehen. Besondere Bedeutung haben dabei die virtuellen Welten von *Online Games* und die virtuellen Charaktere der Spieler (*Avatars*), die eine virtuelle Repräsentation des Community-Mitglieds darstellen.
- **Hybride Communities** (*Hybrid*): enthalten verschiedene Arten und gemischte Gemeinschaften.

Für diese Arbeit relevant ist hauptsächlich der Bereich der Communities, die rund um virtuelle Welten von Spielen entstehen. Diese werden auch als "Gaming-Communities" bezeichnet.

3.3.2 Gaming-Communities

Eine Community ist also eine Interessengemeinschaft, die sich durch gemeinsames Agieren und Austauschen bildet. Das eigentliche Community-Leben findet aber nicht auf makroskopischer Ebene statt. Die Basis der Community bilden ihre einzelnen Mitglieder. Bei den Gaming-Communities handelt es sich um Online-Communities, die rund um Spiele entstehen und deren Basis das gemeinsame Interesse an dem jeweiligen Spiel ist. Sie bieten eine Umgebung, in der ihre Mitglieder miteinander kommunizieren, interagieren und Ideen und Inhalte untereinander austauschen können. Teilnehmer von Gaming-Communities sind natürlich die Spieler, aber auch Entwickler, Publisher oder sonstige Interessenten wie Anbieter von Kommunikationssoftware, Webportalen oder Server-Hosts. Durch ihre Interaktion generieren die Mitglieder Inhalte für die Community, beeinflussen ihr Wachstum und sind für die Weiterentwicklung, bzw. Evolution, der Community verantwortlich. Letztendlich ist auch das "Wohl" des Spiels mit seiner Community verknüpft. Besonders bei MMOGs hängt der Erfolg oder Misserfolg eines Spiels vor allem von der Community und insbesondere von dem Zusammenhalt innerhalb der Communtiy ab [Str07].

Eine Community ermöglicht eine Vielfalt von Aktivitäten. Dabei spielen Dokumentation und Kommunikation eine wichtige Rolle. Speziell in Gaming-Communities können die Mitglieder zum einen innerhalb des Spiels (*in-game*), aber auch außerhalb (*out-game*) aktiv werden. Die Aktivitäten innerhalb von Spielen können je nach Spiel sehr unterschiedlich sein, da es Gaming-Communities zu allen Arten von Spielen gibt. Außerhalb der Spiele sind die Aktivitäten im Allgemeinen jedoch sehr ähnlich, unabhängig vom jeweiligen Spiel. Dabei sind die Mitglieder von Gaming-Communities hauptsächlich in den folgenden Bereichen aktiv, wobei eine Vielzahl an Applikationen, Organisationsformen und Artefakten entstehen.

- **Informations- und Wissenssammlung**: Datenarchive, Datenbanken, Wikis, ...
- **Hilfestellungen**: Guidelines, Foren, Howto's, Chatrooms, ...
- **Auswertungen und Feedback**: Replays, Foren, Beta Tests, ...
- **Planung und Entwicklung von Taktiken**: Kommunikationstools, Foren, Kollaborationstools, ...
- **Organisation**: Webpages (Clanpages), PHP-Anwendungen, Kommunikationstools, ...
- **Sportliche Wettkämpfe (e-sport)**: Clans, Turniere, Ligen (ESL), Weltmeisterschaften (WCG), ...
- **Veränderungen und Erweiterungen**: Spielmodifikationen (Mods), Addons, Macros, ...
- **Weitergehende Aktivitäten**: In-game Filme (Machinima), Speedgaming, Item Handel, ...

Es wird ersichtlich, dass der Bereich der Aktivitäten von Gaming-Communities sehr breit gefächert ist und somit ein weites Spektrum an Beschäftigungsmöglichkeiten für Community-Mitglieder bietet. D.h. jeder kann seinen Vorlieben entsprechend innerhalb der Community tätig werden. Die Relevanz von Gaming-Communities für die einzelnen Mitglieder zeigt auch die Studie von Hsu [Hsu05]. Demzufolge fragen 34% aller Spieler im Falle eines Problems im Spiel zuerst einen Freund nach Rat, 32% suchen Hilfe in Online-Communities und nur 17% erhoffen sich eine Lösung ihrer Probleme seitens des Entwicklers.

3.3.3 MOG Gaming-Communities - Soziale Vernetzung

MOGs zeichnen sich durch das gemeinsame Spielen mit oder gegen andere Spieler aus. Das Spielen erfolgt somit nicht isoliert durch einzelne Personen. Es bilden sich Gruppen von Spielern, die auf verschiedene Arten miteinander interagieren und kooperieren (z.B. sich austauschen, kommunizieren oder Wettkämpfe austragen). Diese Interaktion und Kooperation bleibt dabei nicht nur auf das Spiel beschränkt.

Um das Spiel bilden sich Gemeinschaften, die sich durch das gemeinsame Interesse am Spiel auszeichnen (Gaming-Communities) Das Zusammenschließen von Spielern zu Gruppen und die Organisation von gemeinsamen Aktivitäten haben sich über die Jahre weiterentwickelt. Heutzutage sind Gaming-Communities ein fester Bestandteil im Bereich der MOGs. Das Engagement der Spieler für ihre Spiele ist im Allgemeinen sehr groß und der Austausch und die Kommunikation zwischen den Spielern sind essentielle Elemente von MOGs. Spieler unterstützen sich gegenseitig, tauschen ihre Erfahrungen aus, treffen sich zu Wettkämpfen, organisieren sich in festen Gruppen und erstellen ganze Wissenssammlungen über ihre Spiele.

Die meisten MOGs ermutigen und unterstützen die Interaktion zwischen Spielern, da sie erkannt haben, dass diese essentiell für die virtuelle Welt ist. Jedoch ist die angebotene Unterstützung innerhalb der Spiele nicht ausreichend. Somit gestaltet die Community selbst ihr Umfeld und auch die Inhalte, die sie als interessant oder hilfreich erachtet. Die Ausprägungen, die dabei entstehen sind vielfältig.

Dieser Abschnitt stellt die soziale Interaktion von Spielern und die Bedeutung der sozialen Erfahrung für die Spieler vor. Verschiedene Untersuchungen in der Literatur zeigen (wie zum Beispiel bei [DYNM07], [Man03] oder [SJLK04]), warum Gaming-Communities im Spielumfeld außerhalb des eigentlichen Spiels so aktiv sind, und dass der Bedarf der Spieler an unterstützenden Mechanismen und Tools mit den heute verfügbaren Mitteln nicht ausreichend gedeckt werden kann.

"*The social side of gaming*"

Alle Multiplayer Spiele zeichnen sich durch Interaktion der Spieler untereinander als zentrales Element aus. Durch MOGs wird darüber hinaus die Kommunikation über große Distanzen und mit vielen anderen ermöglicht. "*Gameplay itself is a form of social communication*" [SZ04]. Alle Spiele sind kooperativ auf einer fundamentalen Ebene (systemische Kooperation). Vor allem MOGs unterstützen darüber hinaus die direkte Kooperation zwischen Spielern, indem sie gemeinsame Aktivitäten von Spielergruppen unterstützen. "*Computer gaming has always been a social affair*" [BB04]. Besonders bei MOGs entwickeln sich Communities fast automatisch aus dem Sozial- und Kommunikationsverhalten der Spieler. Da in MOGs "*player-to-player interactions*" [DM04] (Interaktionen zwischen Spielern) und Teamplay wichtige Elemente sind, bilden sich auch schnell feste Gruppen von Spielern, die sich zusammenschließen, um mehr als einmal gemeinsam Aufgaben innerhalb eines Spieles zu lösen. Solche Gruppen werden allgemein als Clan (*team*) oder Gilde bezeichnet oder in manchen Fällen auch mit spielbezogenen Begriffen (zum Beispiel als Allianz oder *Federation*).

Aktivitäten und Organisation solcher Gruppen gehen dann schnell über das eigentliche Spiel und das eigentliche Spielen hinaus und vor allem die Gruppenorganisation findet in der Regel hauptsächlich außerhalb des Spieles statt. Dabei bleibt das soziale Netz (Bekanntschaftsnetzwerk des Spielers, also die Personen mit denen der Spieler sozial interagiert), das ein Spieler bezogen auf ein Spiel aufbaut, nicht auf eine feste Gruppe beschränkt. Die entstehenden Gruppen sind soziale Institutionen, die durch ihre Strukturen die Basis der Gaming-Community bilden [DYNM07].

Das gemeinsame Spielen und die festen Spielergruppen sind ebenso sozial wie Gruppenorganisationen in anderen Bereichen (beispielsweise in Vereinen) und vergleichbar mit einer Manschaftssportart, mit eigenen Teamstrategien und sozialen Normen. Innerhalb dieser sozialen Gruppen ergeben sich eigene Taktiken, Strategien, Stile und Ziele, die aus dem Spiel einen Raum der Sozialisation, Organisation und Vernetzung machen der oft unabhängig vom eigentlichen Spiel ist [WDX+06]. Daher ist die Entwicklung und Verbreitung des elektronischen Sports (e-sports) auch eine logische Konsequenz der MOGs.

Die "anderen Spieler" sind das, was ein MOG ausmacht. Dabei sind die anderen Spieler in ganz unterschiedlicher Art und Weise für das MOG und seine Spieler von Bedeutung und es finden verschiedene Formen sozialer Interaktion statt. Die Formen der sozialen Interaktion innerhalb der Spiele lassen sich nach Art der dabei stattfindenden Kooperation unterscheiden. Verschiedene Autoren (zum Beispiel Nardi und Harris [NH06], Smith [Smi05] oder Ducheneaut et al. [DYNM06]) haben sich mit der sozialen

Interaktion in MOGs beschäftigt, jedoch sind sowohl die verwendete Betrachtungsweise als auch das Vokabular dieser Arbeiten nicht einheitlich. Abbildung 3.15 ordnet die verschiedenen Sichtweisen und fasst sie zusammen.

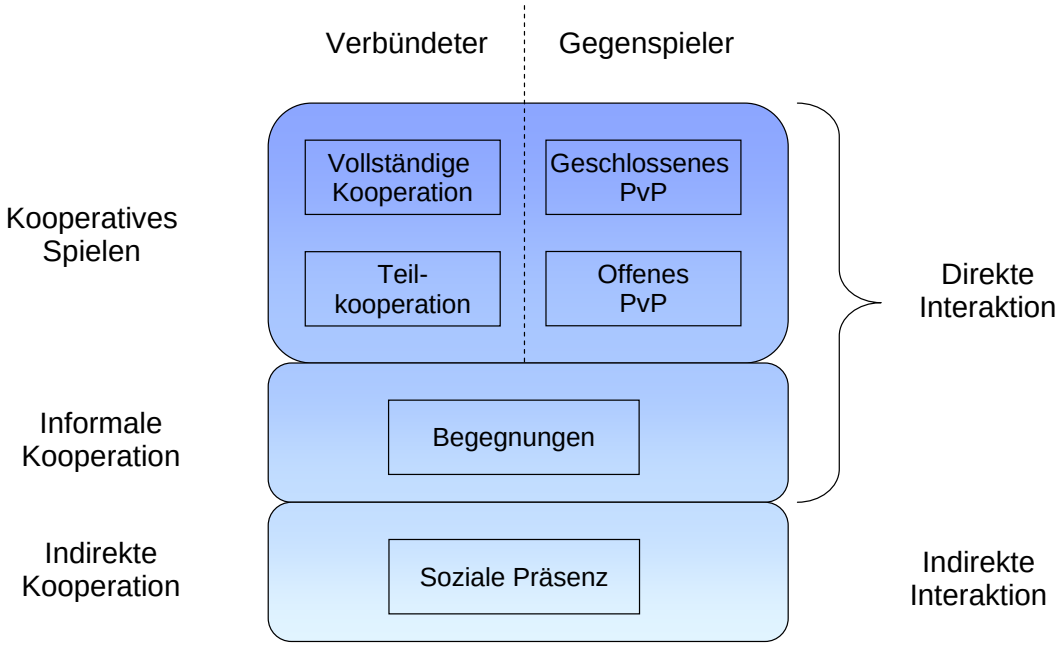

Abbildung 3.15: Formen der sozialen Interaktion

Direkte soziale Interaktion in einem Spiel findet immer dann statt, wenn mindestens zwei Spieler innerhalb des Spieles direkt miteinander interagieren. Dabei ist die soziale Interaktion des **kooperativen Spielens** am stärksten, vor allem, wenn die Interaktion mit anderen Spielern sowohl geplant als auch zielgerichtet ist und unmittelbare Konsequenzen hat. Dabei können die anderen Spieler entweder die Verbündeten des Spielers (Team) aber genauso auch seine Gegenspieler (PvP) sein. Im ersten Fall spricht man von "*vollständiger Kooperation*" (*Coop*), dabei müssen beispielsweise komplexe Manöver abgestimmt, koordiniert und zusammen ausgeführt werden. Um dies erfolgreich umsetzten zu können, sind ein geteiltes Verständnis der taktischen Vorgehensweisen, eine geteilte Analyse der Situation, eine Absprache und das Einstellen auf die Fertigkeiten und Fähigkeiten der Mitspieler erforderlich. Im zweiten Fall handelt es sich um geplante Wettkämpfe gegen andere Spieler "*geschlossenes PvP*" (*Closed PvP*, zum Beispiel in einer speziellen Arena), dabei muss sich der Spieler sowohl auf seine Mitspieler als auch analog auf seine(n) Gegenspieler einstellen. Etwas schwächer ist die sogenannte "*Teilkooperation*" (*Semi-Coop*). Auch diese Form der Kooperation wird durch die Spieler geplant. Die Spannweite der Kooperationsmöglichkeiten ist hier jedoch relativ groß und beinhaltet gemeinsame Unternehmungen, von Spielern organisierte in-game Events genauso wie den Handel mit Gegenständen oder Dienstleistungen, aber auch weniger koordinierte Zusammentreffen mit gegnerischen Spielern, "*offenes PvP*" (*Open PvP*, beispielsweise auf großen Schlachtfelder oder im RvR - *Realm vs. Realm* [43]). **Informale Kooperation** findet ungeplant statt und wird durch zufälliges Aufeinandertreffen "*Begegnungen*" von Spielern in der Spielwelt initiiert. Ausprägungen dieser Art der Kooperation sind beispielsweise das spontane Helfen ("*strangers in the fight*" [NH06]), das unaufgeforderte Verteilen von positiven Effekten an fremde Mitspieler (*buff's*), das Beantworten von Fragen im allgemeinen Chat oder auch das spontane gemeinsame Tanzen ("*random acts of fun*" [NH06]). Auch hier gibt es wieder die Möglichkeit, dass der Interaktionspartner kein Verbündeter, sondern ein Gegenspieler ist. Informale, ungeplante Kooperationen mit anderen Spielern erzeugen zusätzliche Tiefe im Spiel, während die Spieler in der virtuellen Welt unterwegs sind.

Indirekte Kooperation oder **indirekte Interaktion** beschreibt die "*soziale Präsenz*", die das MOG vermittelt, beispielsweise einfach dadurch, dass in einer Stadt innerhalb des Spiels andere Spieler präsent sind oder dadurch, dass in einem Chat, der unabhängig von der Lokation des Spielers in der virtuellen Welt ist, andere Spieler kommunizieren. Über die direkte soziale Interaktion hinaus stellen die anderen Spieler eine Publikum dar. Die anderen Spieler haben Schlüsselrollen, die das Spielen in einer gemeinsamen Welt erst ausmachen, sie geben dem Spieler das Gefühl nicht alleine zu sein, sie sind sowohl Zuschauer wie auch gewollte oder ungewollt Darsteller ("Publikum und Schauspiel"). Besonders MMORPGs sind Spiele, in denen es um die Reputation eines Spielers geht und um das Ansehen innerhalb einer Gruppe. Wichtige Faktoren sind dabei Sehen und Gesehen werden, die Möglichkeit seine Errungenschaften zu präsentieren und das Gefühl, Teil von etwas Größerem zu sein. Das kann aber nur über eine adäquate Interaktion mit dem Publikum erfüllt werden.

Diese Erkenntnisse gehen über Spiele an sich hinaus. Onlinewelten im Allgemeinen brauchen, um erfolgreich zu sein, sowohl eine Unterstützung der direkten Interaktion zwischen ihren Teilnehmen als auch eine Unterstützung der indirekten Interaktion (soziale Präsenz).

Auf einer etwas abstrakteren Ebene findet zusätzlich noch die übergeordnete Interaktion (*higher-level interaction*) statt [Man02]. Diese hat Einfluss auf die Gaming-Community und ihre Entwicklung. Sie beinhaltet Gruppendynamik, kognitive Prozesse, motivationale Aspekte und stille Kommunikation.

Spielumfeld

Damit die virtuelle Welt auch "funktionieren" kann, muss sie eine Unterstützung der benötigten Interaktionsformen anbieten und die Bildung von Communities fördern. Dabei besteht eine Vielzahl an Herausforderungen, wie die Organisation und Verwaltung von festen Spielergruppen oder den Spielern die Möglichkeit zu geben sich darzustellen und teilzuhaben.

Dabei reicht jedoch die Unterstützung der sozialen Interaktion, die die Spiele selbst bieten, bei weitem nicht aus, sodass die Spieler andere Lösungen und externe Unterstützung suchen und aufbauen [Man03]. Die Spieler benutzen beispielsweise zusätzliche Tools, die das Management von Gruppen unterstützen, um funktionierende Gruppen, die über einen längeren Zeitraum bestehen bleiben, unterhalten zu können [DYNM06]. Aus diesem Grund entwickeln die Spieler eigene Tools oder verwenden spielexterne Tools für Funktionen oder Interaktionsformen, die im Spiel nicht angeboten werden [Koi03]. Sie verwenden externe Möglichkeiten, um sich auszutauschen, Aktivitäten zu planen und zu koordinieren, ihr im und über das Spiel angesammeltes Wissen weiterzugeben und sich selbst den anderen Spielern zu präsentieren. Dies Belegen auch Erhebungen über das Verhalten von Spielern, siehe [SJLK04]. Somit entsteht außerhalb des Spiels gemeinsam erarbeitetes Wissen (*Community Knowledge*) [Tay03] und die Grenzen der eigentlichen Spielwelt erweitern sich.

Alle diese Aktivitäten finden somit hauptsächlich außerhalb der Spiele unter Verwendung verschiedener Tools und Applikationen statt, etwa durch den Aufbau von Datenarchiven mit gewaltigen Datensammlungen (thottbot [58]), das Erstellen von Guidelines in Wikis (Guildwiki [77]), durch Forendiskussionen über Taktiken, durch Community-Seiten (gamona.de [76]), Clanseiten oder Blogs mit denen Clans sich und ihrer Mitglieder präsentieren (mTw [41]). Oder durch die Verwendung spezieller Werkzeuge für Teamwork und Kooperation (geteilte Kalender oder Sprachkommunikationstools, wie Teamspeak [57]). Dadurch entsteht eine Art virtueller Raum um das Spiel herum, der Treffpunkt, Diskussionsplattform und Wissenssammlung ist. Dieser virtuelle Raum, der alle Aktivitäten beinhaltet, die über das reine Spielen hinausgehen, sich aber auf das Spiel beziehen, ist das Umfeld des Spiels (*Game Environment*), siehe Abbildung 3.16.

Solch ein Spielumfeld hat jedes MOG. Es entfaltet sich im Internet außerhalb des Spiels und die Gaming-Community erschafft und pflegt dieses Umfeld mithilfe von Tools und Anwendungen, die von der Spieleindustrie, Drittanbietern oder der Community selbst bereitgestellt werden. Das Spielumfeld beinhaltet all diese verschiedenen Internet-Applikationen und Aktivitäten wie Modifikationen, Wissens-

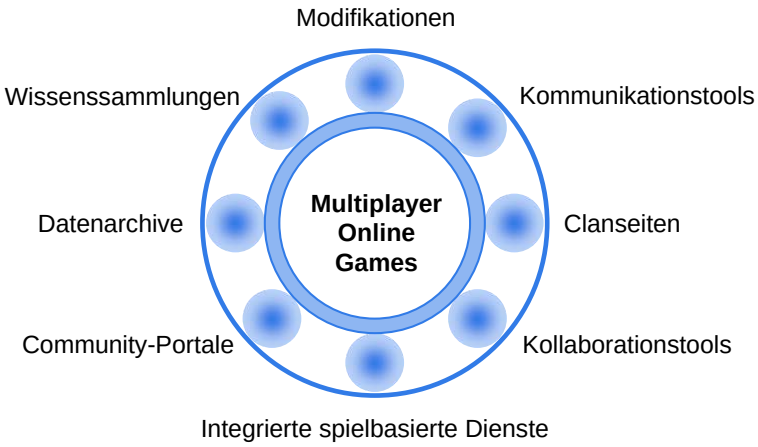

Abbildung 3.16: MOG Umfeld

sammlungen, Datenarchive, Community-Portale, Kommunikationstools, Clanseiten, Kollaborationstools und integrierte spielbasierte Dienste.

Dabei ist das kreative Potenzial der Gaming-Communities enorm hoch. Die Kreativität zeigt sich beispielsweise in den diversen Artefakten, die rund um die Spiele entstehen. Neben dem Spielen, der Beschäftigung mit den Spielinhalten und der Anpassung von Spielfunktionen werden MOGs von der Community wiederum als Grundlage für neue Spielkonzepte (wie beim *Speedgaming*) oder eigene Geschichten verwendet (wie bei Machinima, Spielecomics oder spielbasierten Hörspielen). Machinima sind von der Community erstellte Filme, die innerhalb einer Spielwelt "produziert" werden, aber eine eigene Geschichte erzählen. Eine Sammlung von Machinima, basierend auf Spielen wie *Half-Life* [64], *World of Warcraft* [11] oder auch *Super Mario*-Spielen, bietet beispielsweise die Webseite Machinima.com [35]. Ein spielbasiertes Hörspiel ist die Allimania-Reihe [54], die in *World of Warcraft* angesiedelt ist und auch teilweise innerhalb des Spiels "verfilmt" wird. Spielecomics gibt es in diversen Ausprägungen. SO finden sich unter anderem direkt auf einem bestimmten Spiel oder Spielszenario basierende Comics, die entweder automatisch generiert [CTC09] oder von Spielern erstellt werden (Shakes & Fidget [48], Looking for Group [52]). Auch Comics über Gamer und das Spielen im allgemeinen (Ctrl+Alt+Del [15]) werden verfasst.

Community-Beteiligung

Die Aktivitäten der Gaming-Communities zeigen, dass die Spieler nicht mehr nur konsumieren, sondern auch an der Entwicklung der Spiele teilnehmen wollen. Neben den Aktivitäten im Spielumfeld wollen die Gaming-Communites sich aktiv und kreativ an ihren Spielen beteiligen und direkten Einfluss auf ihre Spiele nehmen, sei es auf das Design, die Darstellung oder die Kontrolle (beispielsweise durch Befragungen, Beta-Tests oder aktive Einbindung). Das gilt in besonderem Maße für MMORPGs, da diese sich ständig weiterentwickeln. Die Spieler wollen an diesem Entwicklungsprozeß mitwirken und ihre eigene Kreativität und eigene Ideen einbringen. Die aktive Einbindung, wie etwa durch die Einbeziehung von Inhalten, die von den Spieler erstellt wurden (*player created content*), stärkt die Community des Spiels [Koi03]. Zusätzlich können Spiele, die Möglichkeiten zur Beteiligung der Spieler bieten, von der kreativen Partizipation der Community enorm profitieren.

Die Spieleindustrie bietet den Gaming-Communities bereits unterschiedliche Formen der Beteiligung an, wobei die Stärke der Einflussmöglichkeiten, die die Spiele bieten, variiert. So bieten einige FPS beispielsweise die Möglichkeit das Spiel zu modifizieren (zum Beispiel *Unreal Tournamant* [Rei99]) oder enthalten spezielle Editoren, die es den Spielern erlauben, Spielinhalte zu verändern oder neu zu erschaffen. Somit wird der Community beispielsweise ermöglicht, neue Karten (*Maps*) zu designen und

damit die Vielfalt des Spieles anzureichern. Eine etwas andere Form der Community-Einbindung entsteht dadurch, die Beteiligung der Community bereits im Spielkonzept vorzusehen. So wurden Spielen wie *SPORE* [36] oder *LittleBIGPlanet* [38] entworfen mit der Idee die Game-Community aktiv an der Erstellung von Inhalten (*player created content*) für die Spiele zu beteiligen. Dabei werden die kreativen Elemente der Spiele dazu benutzt, die von den Spielern erstellten Komponenten an andere Spieler weiterzugeben, damit diesen Diversität geboten werden kann, ohne dass der Entwickler selbst tätig werden muss. Beispiele für die unterschiedlichen Möglichkeiten der aktiven Community-Einbindung sind:

- Meinung einholen, bei der Fehlersuche beteiligen, z.B.: Befragungen, Abstimmungen, Beta-Tests, Bug-Report
- Inhalte aufnehmen, z.B.: Anregungen der Spieler umsetzten, *player created content*
- Spielerweiterungen zulassen, z.B.: AddOns
- Spielveränderungen zulassen, z.B.: Modifikationen (*Mods*)

Eine **Modifikation** (*Mod*) ist eine von Spielern veränderte Version eines Spiels. Diese Veränderung kann bestehende Spielinhalte in ihrem Erscheinungsbild (Aussehen, Klang, etc.) oder in ihrer Funktionalität (Spielabläufe) betreffen und bis zu einer kompletten Umwandlung (*total conversion*) des Spieles in ein "anderes" Spiel (Spielwelt, Spielmechanik, Spielziel, usw.) reichen. Typische Mods, die das Erscheinungbild verändern, beinhalten veränderte Maps, Grafiksets oder Audiopakete. Änderungen der Funktionalität sind zum Beispiel eine veränderte Spielphysik und verändertes Verhalten von Gegenständen im Spiel. Die Funktionalitätsveränderungen sind oft modular und können miteinander kombiniert werden (man bezeichnet sie auch als *Mutator*). Andere typische Modifikationen verändern den Spielmodus (Beispiele typischer Modi sind: *Capture the Flag*, *Team-Deathmatch* und *Domination*). Eine komplette Umwandlung verändert alle Elemente des Spiels, sodass ein neues Spiel entsteht. Das prominenteste Beispiel ist das Spiel *Counter-Strike* [65], das ursprünglich als eine Modifikation das Spiels *Half-Life* [64] von Mitgliedern der Community erstellt wurde und so erfolgreich und beliebt war, dass die Entwickler von *Half-Life* (Valve [63]) *Counter-Strike* in ihr Repertoire aufnahmen und eine offizielle Retail-Version von *Counter-Strike* auf den Markt brachten sowie eine Nachfolgeversion (*Counter-Strike: Source* [66]) entwickelten. Die angebotenen Modifikationsmöglichkeiten werden von der Community genutzt und so entstehen hunderte neuer Mods und Maps zu den unterschiedlichsten Thematiken, die teilweise kreative Kunstwerke darstellen [WBB02].

Weitere Möglichkeiten, Veränderungen oder Erweiterungen an Onlinespielen durchführen zu können, sind sogenannte **Addons**. Mit Addons können Spiele, die eine entsprechende API bereitstellen, erweitert werden. Mit diesen Erweiterungen können bestimmte Aspekte im Spiel angepasst werden, das eigentliche Spielgeschehen bleibt jedoch unbeeinflusst. Ein Beispiel sind Addons für *World of Warcraft*, die in der Skriptsprache Lua [30] programmiert werden können. Typische Addons ermöglichen eine individuelle Anpassung der Benutzerschnittstelle (*User Interface* - UI) oder eine verbesserte oder erweiterte Anzeige von Informationen. Der Erfolg des Spiels *World of Warcraft* [12] zeigt auch eindrucksvoll, welchen Einfluss und welches Potenzial solch eine Beteiligung der Community hat. *World of Warcraft* kann als positives Beispiel der Community-Beteiligung unter den MMORPGs betrachtet werden. Durch die Möglichkeit, mit Addons Anpassungen des Benutzerinterfaces durchführen zu können, kann es an die Bedürfnisse des Benutzers angepasst werden. Durch die API, die vom Spiel bereitgestellt wird, können bestimmte in-game Informationen außerhalb des Spiels abgefragt werden. Auch wenn die Informationen, die abgefragt werden können, beschränkt sind und die Anpassungen, die vorgenommen werden können, begrenzt sind, nimmt die Communitiy diese Einflussmöglichkeiten dankbar an und setzt sie in vielfältiger Weise um. Dies geschieht beispielsweise durch die Erstellung von spielbezogenen Datensammlungen und selbst kreierten Interface-Erweiterungen. Alleine die Addon-Sammlung des Internet Portals curse.com [18] enthält über 4000 von der Community erstellte Addons für *World of Warcraft*. Von der Kreativität der Community profitieren wiederum sowohl die Spieler, als auch das Spiel selbst.

Oft werden die Ideen beliebter und nützlicher Addons von den Spielentwicklern aufgegriffen und fest in das Spiel integriert.

4 Virtueller Kontext in virtuellen Welten

Wie zuvor in Kapitel 3.2.1 dargestellt, sind virtuelle Welten (*Networked Virtual Environments*) simulierte Realitäten, die in der Regel physikalischen Gesetzmäßigkeiten wie Gravitation und anderen Prinzipien der realen Welt folgen, jedoch meist in einem Fantasy- oder Science Fiction- Szenario angesiedelt sind. Wichtige Eigenschaften dieser virtuellen Welten sind Repräsentationen von Benutzern innerhalb der virtuellen Umgebung, vielfältige Interaktionsmöglichkeiten und assoziierte Communities. Auch innerhalb einer virtuellen Welt besitzt der Benutzer einen Kontext, analog zu seinem Kontext in der realen Welt. Eine gezielte Betrachtung dieses virtuellen Kontexts eröffnet neuartige Anwendungskonzepte und Verwendungsmöglichkeiten. Dabei ergeben sich folgende grundsätzliche Fragestellungen:

- Definition: Was ist virtueller Kontext?
- Beschreibung: Wie kann virtueller Kontext beschrieben werden?
- Verwendung: Welche Verwendungsmöglichkeiten und -konzepte gibt es für virtuellen Kontext?

Im Rahmen dieser Arbeit wurde eine Untersuchung von virtuellem Kontext anhand einer umfangreiche Analyse virtueller Welten im Allgemeinen und MOGs im Speziellen durchgeführt. Im Folgenden werden die Ergebnisse dieser Analyse, die zur Beantwortung der drei Grundsatzfragen beitragen, sowie die daraus abgeleiteten Modelle und Schlußfolgerungen dargestellt. Als weitere Ergebnisse werden im zweiten Abschnitt die Definition von virtuellem Kontext formuliert und die Eigenschaften virtueller Parameter dargestellt. Im dritten Abschnitt wird die entwickelte Beschreibung virtueller Kontextinformationen vorgestellt und im vierten Abschnitt werden die Verwendung von virtuellem Kontext dargestellt sowie die virtuell kontextbasierten Dienste (*virtual context based services*, VCBS) eingeführt und definiert. In den darauf folgenden Kapiteln wird dann die Umsetzung von virtuell kontextbasierten Diensten im Anwendungsszenario dieser Arbeit behandelt.

4.1 Interaktion in virtuellen Welten

Für das grundlegende Verständnis von virtuellem Kontext ist eine detaillierte Analyse der virtuellen Welten nötig. Virtuelle Welten zeichnen sich durch die Interaktionsmöglichkeiten aus, die durch die Schnittstellen der virtuellen Welt ermöglicht werden. Dabei entstehen unterschiedliche Möglichkeiten, in und mit der virtuellen Umgebung zu interagieren. Die Interaktionen, die in Bezug auf die virtuelle Welt durchgeführt werden, sind ein zentraler Aspekt virtueller Welten. Die Untersuchung der Interaktionsmöglichkeiten in der virtuellen Welt und ihrer Einflussfaktoren ist aus diesem Grund ein zentraler Aspekt der durchgeführten Analyse. Das Spielen in der Gruppe, als wichtiger Spielbestandteil von MOGs, beinhaltet ein hohes Maß sowohl an Interaktion als auch an Dynamik. Bei der durchgeführten Analyse virtueller Welten wurde auch die Gruppeninteraktion detailliert betrachtet.

Die im Folgenden vorgestellten Analyseergebnisse beinhalten die detaillierte Untersuchung der Interaktionsmöglichkeiten sowie die Betrachtung der verschiedenen Schnittstellen (*Interfaces*). Das aus der Analyse resultierende generische Modell der Interaktion in virtuellen Welten wird vorgestellt. Im Anschluss werden die Ergebnisse der Untersuchung von Interaktion beim Gruppenspiel genauer dargestellt sowie die entwickelte Interaktionsmodellierung kurz vorgestellt.

4.1.1 Interaktion und Interfaces

Interaktion ist das grundlegende Konzept in Spielen. "*Games are one of the most ancient forms of desingned human interactivity...*" [SZ04]. Spielen bedeutet, Entscheidungen zu treffen innerhalb eines Spielesystems das konzipiert wurde, um Aktionen umzusetzten und Resultate zu erzeugen. Jede Aktion resultiert in einer Änderung bezüglich des Gesamtsystems. Dieser Prozess aus Aktion und Resultat entsteht durch die Interaktion des Benutzers mit dem System. "*The word "interactivity" isn't just about giving players choices; it pretty much completely defines the game medium.*" (Warren Spector in [SZ04]). Interaktion findet dabei auf allen Ebenen statt, von der formalen Interaktionen mit den virtuellen Objekten der Spielwelt, über die soziale Interaktion der Benutzer, bis hin zur kulturellen Interaktion des Spieles mit der Welt jenseits der Spielgrenzen.

Es gibt verschiedene Interfaces, die Interaktionen in einer virtuellen Welt oder bezogen auf eine virtuelle Welt ermöglichen. Übliche Interfaces sind die virtuelle Repräsentation der Spieler (*virtueller Charakter*), *Chat* (Text-Chat und Voice-Chat) und das abstrakte *Community*-Interface.

- **Virtueller Charakter**: In virtuellen Welten ist der virtuelle Charakter das primäre Benutzerinterface. Mit seinem virtuellen Charakter interagiert der Benutzer in und mit der virtuellen Umgebung und den virtuellen Charakteren der anderen Benutzer der virtuellen Welt.

- **Chat**: Mit einem Chat (die meisten virtuellen Welten haben einen integrierten Text-Chat) kann der Benutzer direkt mit einem oder mehreren anderen Benutzern per Text interagieren. Besonders für die Sprachkommunikation werden zusätzliche Voice-Chat Tools eingesetzt. Diese Tools werden parallel zu der virtuellen Welt verwendet und sind weit verbreitet unter den Benutzern (z.B. TeamSpeak [57]).

- **Community**: Innerhalb der Community interagiert der Benutzer mit vielen anderen Benutzern der virtuellen Welt, jedoch findet diese Interaktion zum größten Teil außerhalb der eigentlichen virtuellen Welt statt, z.B. über Foren, spezielle Community Portale (wie buffed.de [17]) oder andere Internet-Applikationen (siehe auch 3.3.2).

Abhängig von den verschiedenen Interfaces, die einem Benutzer zur Verfügung stehen, ergeben sich für den Benutzer unterschiedliche Interaktionsmöglichkeiten. Jedes Interface hat verschiedene Charakteristika. Die Untersuchung der verschiedenen Interfaces hat ergeben, dass es Eigenschaften gibt, anhand derer die ermöglichten Interaktionen beschrieben und kategorisiert werden können. Die Interaktionen, die durch die verschiedenen Benutzer-Interfaces ermöglicht werden, können anhand der *Interaktionsprozedur*, des *Interaktionsortes*, des *Interaktionspartners* und des *Interaktionstyps* eingeteilt werden:

- **Interaktionsprozedur**: Der Typ der Interaktionsprozedur bestimmt den Ablauf der Interaktion. Bei einer *synchronen* Interaktion hat der Benutzer unmittelbaren Kontakt mit seinem Interaktionspartner (einem virtuellen Objekt oder einem anderen Benutzer). Bei einer *asynchronen* Interaktion gibt es eine vermittelnde Plattform, sodass die Interaktionspartner nicht gleichzeitig erreichbar sein müssen (wie bei einem Forum).

- **Interaktionsort**: Der Interaktionsort spezifiziert, wo die virtuelle Interaktion stattfindet. Findet die Interaktion innerhalb der virtuellen Welt statt, so wird sie als *in-game* Interaktion bezeichnet. Interaktionen, die mit der virtuellen Welt zu tun haben oder in Bezug dazu stehen, jedoch außerhalb der virtuellen Welt stattfinden, werden als *out-game* Interaktion bezeichnet.

- **Interaktionspartner**: Der Interaktionspartner bezeichnet, mit wem der Benutzer interagiert. Eine Interaktion kann direkt mit der virtuellen Welt stattfinden inklusive virtueller Objekte oder computergesteuerter Charaktere (NPCs). In diesem Fall handelt es sich um einen *virtuellen* Interaktionspartner. Wenn ein anderer menschlicher Benutzer in die Interaktion involviert ist, auch wenn dieser durch seinen virtuellen Charakter repräsentiert wird, wie z.B. bei der in-game Interaktion mit dem

virtuellen Charakter eines anderen Benutzers der virtuellen Welt, so ist der Interaktionspartner *real*.

- **Interaktionstyp**: Der Interaktionstyp kennzeichnet die Art der durchgeführten Interaktion. Handelt es sich dabei um eine konkrete Interaktion, wenn also ein oder mehrere andere Benutzer oder die virtuelle Welt sind Ziel der Interaktion, so gibt es zwei Interaktionstypen. Der erste Typ ist eine *Aktion in der virtuellen Welt*. Die Interaktion wird virtuell (durch den virtuellen Charakter) ausgeführt, wie zum Beispiel das Laufen mit einem virtuellen Charakter durch die virtuelle Umgebung oder das Öffnen einer virtuellen Kiste. Der zweite Interaktionstyp ist die *Kommunikation*. Sie umfasst Interaktionen, die über Sprache ausgeführt werden (gesprochen oder geschrieben). Darüber hinaus gibt es noch einen weiteren abstrakten Interaktionstyp (ohne festes Interaktionsziel), die *erweiterte Kommunikation*. Die erweiterte Kommunikation beinhaltet den Austausch von *user generated content*, der sich auf die virtuelle Welt bezieht.

Interaktionen können also sowohl innerhalb der virtuellen Umgebung als auch außerhalb davon stattfinden. Den Zusammenhang zwischen dem Interaktionsort und den möglichen Interaktionstypen zeigt Abbildung 4.1.

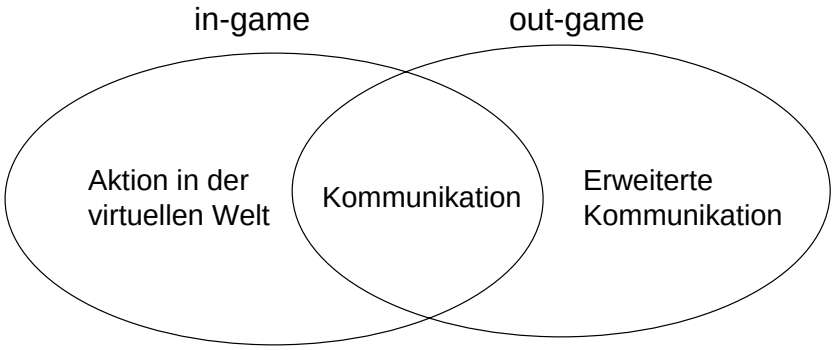

Abbildung 4.1: Zusammenhang zwischen Interaktionsort und Interaktionstyp

Eine komplette Übersicht der Interaktionsinterfaces und Interaktionseigenschaften zeigt Tabelle 4.1. Die erste Zeile zeigt die Eigenschaften für das Interaktionsinterface *virtueller Charakter*, der mit der virtuellen Welt (virtuellen Objekten und NPCs) interagiert. In diesem Fall handelt es sich immer um eine *synchrone* Interaktion, die innerhalb der virtuellen Welt *in-game* mit einem *virtuellen* Interaktionspartner ausgeführt wird und eine *Aktion in der virtuellen Welt* (Aktion idvW) darstellt. Bei allen anderen Formen der Interaktion ist der Interaktionspartner *real*, also ein anderer Benutzer. So auch bei der Interaktion des virtuellen Charakters mit einem oder mehreren virtuellen Charakteren anderer Spieler. Hier kann die Interaktion dann sowohl Aktion als auch *Kommunikation* sein. Beim Interface *Chat* handelt es sich hingegen immer um eine Kommunikation, die synchron stattfindet (ein Chat ist ein synchrones Kommunikationstool), wobei Text-Chat Kommunikation in der Regel innerhalb der virtuellen Welt bereitgestellt wird und Voice-Chat Kommunikation in den meisten Fällen durch externe Tools (*out-game*). Die *Community* als abstrakte Form eines Interfaces kann unterschiedlich instanziiert sein und ermöglicht Kommunikation oder auch *erweiterte Kommunikation*. Diese findet außerhalb der virtuellen Umgebung statt und ist überlicherweise eher *asynchron*. Bei der erweiterten Kommunikation finden anhand von Wissenartefakten, die zwischen Benutzern ausgetauscht, kommentiert und verändert werden, abstrakte Kommunikationsprozesse statt, die einen wichtigen Teil der Community-Interaktion ausmachen. Beispiele für eine erweiterte Kommunikation sind Erstellung von Guidelines über die virtuelle Welt oder die Erstellung von Filmen über in-game Aktivitäten durch Screencapturing.

Tabelle 4.1: Interaktion: Interfaces und Eigenschaften

Interaktion	Prozedur: Synchron / Asynchron	Ort: In-game / Out-game	Partner: Virtuell / Real	Typ: Aktion idvW / Kommunikation / erweiterte Kom.
Interface: virtueller Charakter • interagiert mit virtuellen Objekten und NPC's	Synchron	In-game	Virtuell	Aktion idvW
Interface: virtueller Charakter • interagiert mit virtuellen Charakteren (Mitspieler)	Synchron	In-game	Real	Aktion idvW / Kommunikation
Interface: Text-Chat	Synchron	In-game / (Out-game)	Real	Kommunikation
Interface: Voice-Chat	Synchron	Out-game / (In-game)	Real	Kommunikation
Interface: Community	Asynchron / (Synchron)	Out-game	Real	Kommunikation / erweiterte Kom.

In dieser Übersicht der Interaktionseigenschaften werden nur Basisfunktionalitäten der Interfaces berücksichtigt. Durch Interface-Erweiterungen können zusätzliche Funktionalitäten ermöglicht werden. Zusätzliche Erweiterungen sind jedoch immer spezifisch und abhängig von der betrachteten virtuellen Welt und sind für die allgemeine Betrachtung daher nicht relevant.

4.1.2 Interaktionsmöglichkeiten in virtuellen Welten

Alle Interaktionen, die ein Benutzer in, mit und bezogen auf eine virtuelle Umgebung durchführen kann, werden durch Interfaces ermöglicht. Für die Untersuchung von virtuellen Welten sind Interaktionen, die in Bezug auf die virtuelle Welt durchgeführt werden, ein Schlüsselaspekt (siehe auch [Man00]). Die virtuelle Umgebung in der sich der Benutzer mittels seines virtuellen Charakters aufhält, beeinflusst die Interaktionsmöglichkeiten, da sich auch in der virtuellen Welt die Gegebenheiten und somit die zur Verfügung stehenden Interaktionen je nach Situation ändern. Analog zu der Situation des Benutzers in der realen Welt, befindet sich sein virtueller Charakter in unterschiedlichen virtuellen Situationen, wenn er sich in der virtuellen Welt befindet. Grundlage der Untersuchung von virtuellen Welten ist das Verständnis der Interaktionsmöglichkeiten, insbesondere wie eine Interaktion aus den zur Verfügung stehenden Interaktionsmöglichkeiten ausgewählt wird und welche Faktoren diese Auswahl beeinflussen. Daher wurden die Interaktionsmöglichkeiten in der durchgeführten Analyse detailliert untersucht. Dafür wurden die Interaktionstypen betrachtet, durch die eine Interaktion innerhalb der virtuellen Umgebung ermöglicht wird und ihre Eigenschaften wurden analysiert. Im Folgenden werden die Ergebnisse der zwei Basisinteraktionstypen, *Aktion in der virtuellen Welt* und *Kommunikation*, vorgestellt.

Aktion in der Virtuellen Welt

Interaktionen vom Typ **Aktion in der virtuellen Welt** können nur über das Interface *virtueller Charakter* durchgeführt werden. Der virtuelle Charakter ist die virtuelle Repräsentation des Benutzers in der virtuellen Welt (siehe 3.2.2). Die Erstellung der virtuellen Charaktere ist abhängig von der jeweiligen virtuellen Welt. Die größten Einflussmöglichkeiten und die vielfältigsten Interaktionsmöglichkeiten bieten virtuelle Welten aus dem Rollenspiel-Bereich. Aus diesem Grund werden diese im Folgenden bei der Vorstellung der Analyseergebnisse verwendet.

Ein neu erstellter Charakter besitzt von Anfang an Basisfähigkeiten mit denen er bereits mit der virtuellen Welt und andern virtuellen Charakteren interagieren kann (siehe 3.2.2). Diese *inhärenten Fähigkeiten* bieten dem Benutzer alle essenziellen Interaktionsmöglichkeiten, um in der virtuellen Welt zurecht zu kommen. Viele virtuelle Welten bieten dem Benutzer die Möglichkeit, seinen Charakter weiterzuentwickeln (siehe 3.2.2). Der virtuelle Charakter des Benutzers verbessert sich auf Basis von Erfahrungspunkten (XP - *experience points*), die er beispielsweise für das Durchführen von Aufgaben erhält. Durch das Sammeln von Erfahrungspunkten werden neue Fertigkeiten für den Charakter verfügbar, die er erlernen kann. Diese *erworbenen Fähigkeiten* erweitern die Interaktionsmöglichkeiten des virtuellen Charakters. Somit definieren die inhärenten zusammen mit den erworbenen Fähigkeiten alle Aktionen, die mit diesem virtuellen Charakter in der virtuellen Welt ausgeführt werden können (*prinzipielle Aktionsmöglichkeiten*, siehe Abbildung 4.2).

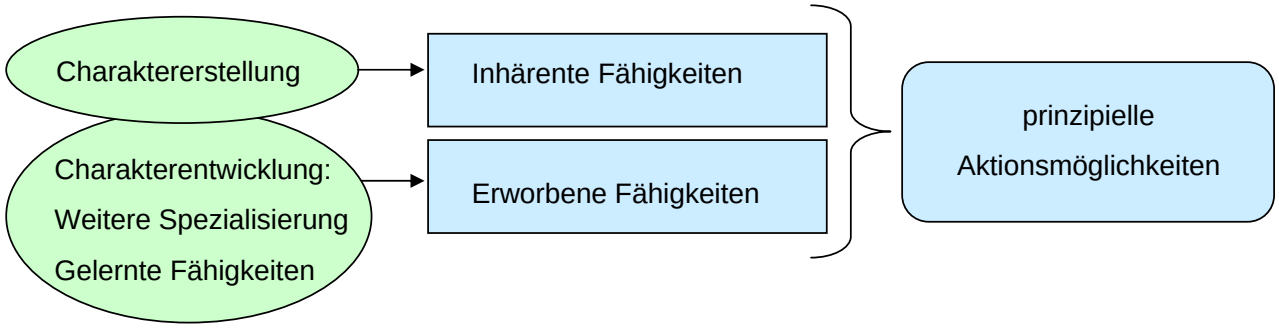

Abbildung 4.2: Prinzipielle Aktionsmöglichkeiten eines virtuellen Charakters in einer virtuellen Welt

Auch der Benutzer des virtuellen Charakters sammelt Erfahrung. Die Spielerfahrung des Benutzers setzt sich jedoch aus allen seinen Erfahrungen, die er bezogen auf das Spiel sammelt, zusammen. Diese beinhalten beispielsweise das Spielen mit unterschiedlichen Charakteren, den Austausch mit anderen Spielern, die Recherche von spielbezogenen Inhalten im Internet oder das Lesen von Artikeln über das Spiel. Der Spieler lernt somit anhand unterschiedlicher Quellen. So können zwei Spieler, die einen virtuellen Charakter mit der selben Anzahl gesammelter Erfahrungspunkte haben, als Personen sehr unterschiedliche Spielerfahrung haben.

Im Folgenden wird nun eine Formalisierung der Erkenntnisse über Aktionen, die in virtuellen Welten durchgeführt werden, vorgenommen. Für Aktionen in virtuellen Welten bilden die *prinzipiellen Aktionsmöglichkeiten* (A_p) die Menge aller möglichen Interaktionen, die in der virtuellen Umgebung durchgeführt werden können. Abbildung 4.3 zeigt das aus der durchgeführten Analyse resultierende Modell der Aktion in virtuellen Welten.

Die Analyse hat ergeben, dass nicht jede Aktion aus der Menge der prinzipiellen Aktionsmöglichkeiten immer durchgeführt werden kann. Wie auch in der realen Welt können in einer virtuellen Welt nicht immer alle prinzipiell verfügbaren Aktionen zu jedem Zeitpunkt und an jedem Ort ausgeführt werden. Auch in virtuellen Welten gibt es verschiedene Arten von Restriktionen. So muss zum Beispiel für die Verwendung einer Schwimm-Fertigkeit auch ein zum Schwimmen geeignetes Gewässer vorhanden sein. Diese Einflussfaktoren zu untersuchen, war der Hauptaspekt der Analyse von Interaktionsmöglichkei-

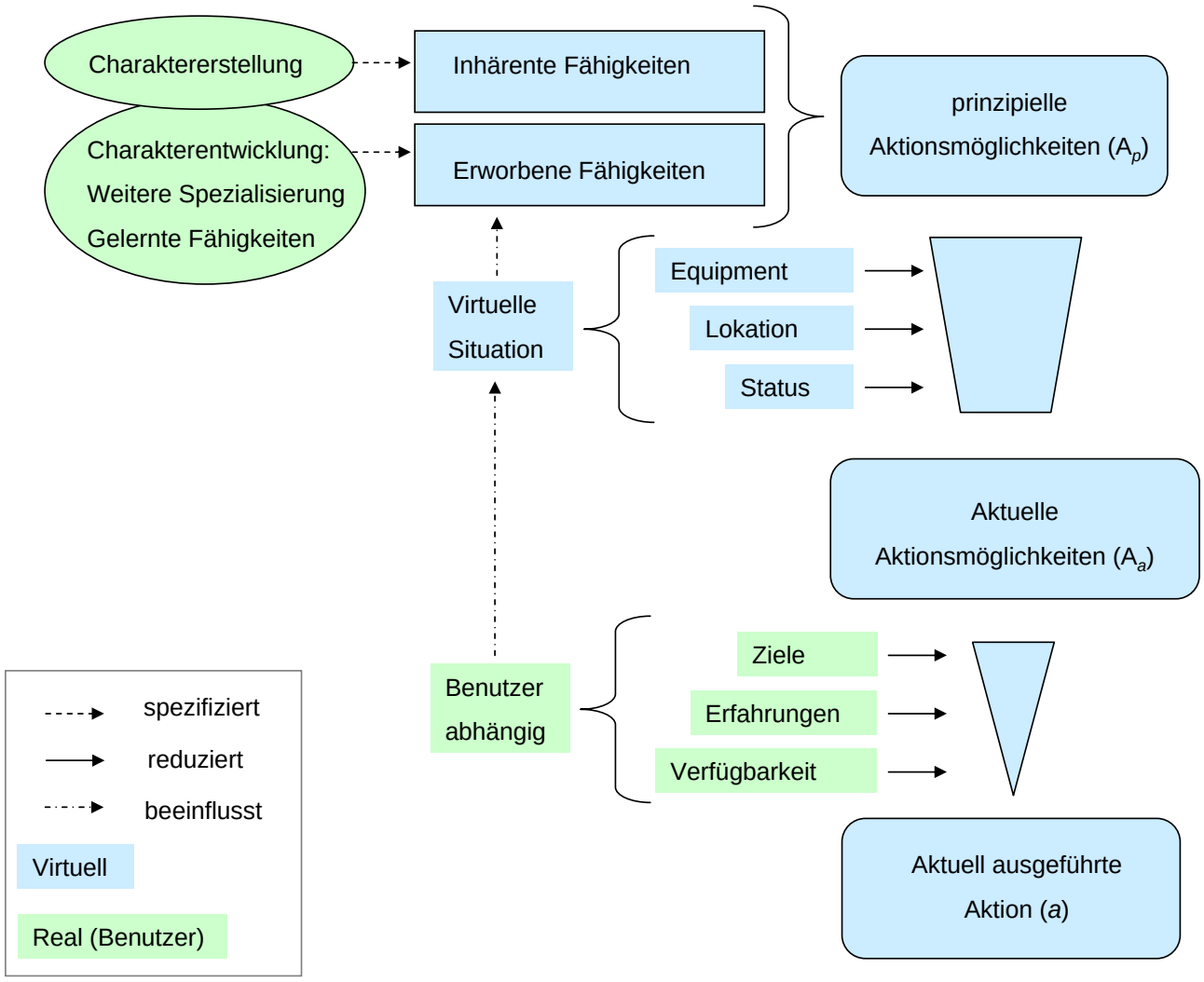

Abbildung 4.3: Modell der Aktion in einer virtuellen Welt

ten in virtuellen Umgebungen. Ob eine Aktion durchgeführt werden kann, ist abhängig von virtuellen Parametern (wie Lokation, Status und Equipment). Diese virtuellen Parameter charakterisieren die *virtuelle Situation* in der sich der Charakter befindet. Abhängig von der virtuellen Situation entsteht eine Teilmenge von Aktionen, die in der aktuellen Situation des virtuellen Charakters gültig sind und somit ausgeführt werden können (*aktuelle Aktionsmöglichkeiten*, A_a).

Ein einfaches Beispiel ist ein virtueller Charakter, der die Fertigkeit hat, virtuelle Kisten zu öffnen. Wenn er nun eine Kiste öffnen möchte, die verschlossen ist, so wird er scheitern, obwohl er eigentlich die Fertigkeit besitzt, sie zu öffnen. Hat er den passenden Schlüssel oder zerstört er das Schloss, dann ist die Kiste unverschlossen und er ist in der Lage, die Kiste zu öffnen.

Aus der Teilmenge der aktuellen Aktionsmöglichkeiten wählt nun der Benutzer die Aktion aus, die er ausführen möchte (*aktuell ausgeführte Aktion*, a). Diese Auswahl ist abhängig von realen Parametern, wie den Erfahrungen, Zielen, etc., die der Benutzer hat (*benutzerabhängig*).

Erweitert man das oben betrachtete Beispiel der zu öffnenden Kiste, so nehmen wir nun an, der Benutzer hat einen passenden Schlüssel und kann außerdem Schlösser von Kisten knacken. Wenn nun der Benutzer seine Fertigkeit im Schlossknacken verbessern will, dann wird er den Schlüssel nicht benutzen. Ist das jedoch nicht sein Ziel und weiß er außerdem aus Erfahrung, dass das Schlossknacken viel mehr Zeit in Anspruch nimmt als das Aufschließen mit einem Schlüssel, den er ja sowieso schon besitzt, dann wird der Benutzer den Schlüssel verwenden, um die Kiste aufzuschließen.

Zusätzlich zum beschriebenen Ablauf im Aktionsmodell gibt es zwei Rückkopplungen, die von Bedeutung sind: Der Einfluss der virtuellen Situation auf den Erwerb neuer Fähigkeiten und die Abhängigkeit der möglichen virtuellen Situationen vom Benutzer.

Die *virtuellen Situationen*, in denen sich ein Charakter bereits befunden hat (seine Historie), können sich auf den Erwerb neuer Fertigkeiten auswirken. Zum Beispiel kann die Fähigkeit, ein schnelles Pferd zu reiten, abhängig davon sein, zuerst einmal reiten zu lernen und dann mit einem langsamen Pferd zu üben, bis das langsame Reiten gemeistert wurde, um danach in der Lage zu sein, ein schnelles Pferd zu reiten.

Benutzerabhängig sind die virtuellen Situationen, in die sich der Benutzer mit seinem Charakter begibt. Denn der Benutzer entscheidet darüber (basierend auf seinen Erfahrungen, Zielen, etc.), welchen Weg durch die virtuelle Umgebung er einschlägt und welche virtuellen Situationen er sucht beziehungsweise meidet.

Welche Fähigkeiten in einer bestimmten Situation benutzt werden können und welche Fähigkeit dann schließlich benutzt wird, hängt von verschiedenen Einflussfaktoren ab. Diese Einflussfaktoren setzten sich aus virtuellen und realen Faktoren zusammen. Zusammenfassend zeigt das Modell, dass die Auswahl einer Aktion, die in einer virtuellen Umgebung ausgeführt wird, in zwei Schritte unterteilt ist. Zunächst schränkt die virtuelle Situation die Menge der prinzipiellen Aktionsmöglichkeiten auf die Menge der aktuell möglichen Aktionen ein und danach wählt der Benutzer basierend auf seinen Erfahrungen und Vorstellungen eine dieser aktuell verfügbaren Aktionen aus.

Das bedeutet:

- Im ersten Schritt entsteht die Teilmenge der aktuellen Aktionsmöglichkeiten A_a aus der Menge aller möglichen Aktionen A_p:

  $$A_a \subseteq A_p$$

- Im zweiten Schritt entsteht die Menge A (die als Element nur die aktuell ausgeführte Aktion a enthält $A = \{a\}$) als Teilmenge von A_a:

  $$A \subseteq A_a$$

- Somit ergibt sich die ausgeführte Aktion a aus der Menge der Aktionsmöglichkeiten A_p:

  $$a \in A \wedge A \subseteq A_a \subseteq A_p \tag{4.1}$$

Kommunikation

Analog zur Analyse der Aktionen in virtuellen Welten ist die Untersuchung der Interaktionen vom Typ **Kommunikation** möglich. Bei der Untersuchung der Kommunikation müssen jedoch, im Gegensatz zu den Aktionen, verschiedene Interfaces betrachtet werden. In einem typischen Szenario gibt es zwei in-game Interfaces für die Kommunikation, den *virtuellen Charakter* und einen *Text-Chat*, sowie ein zusätzliches out-game Interface für die Sprachkommunikation (*Voice-Chat*). Mittels des virtuellen Charakters ist eine *unmittelbare Kommunikation* zwischen virtuellen Charakteren möglich. Das Text-Chat Interface ermöglicht sowohl personenbezogene, *gerichtete Kommunikation* (gerichtet an einen anderen Benutzer oder eine Gruppe), als auch inhaltsbezogene, themen*spezifische Kommunikation* (zu einem bestimmten Thema), siehe Abbildung 4.4. Zusammengenommen definieren die beiden in-game Interfaces die *prinzipiellen in-game Kommunikationsmöglichkeiten* (K_{pi}).

Zu den Kommunikationsformen, die die virtuelle Welt zur Verfügung stellt, werden oft ergänzende Kommunikationsmechanismen, typischerweise ein Sprachkommunikationstool (Voice-Chat Tool), verwendet. Da die Verwendung dieses Interfaces in der Praxis von großer Bedeutung ist und es aktuell als

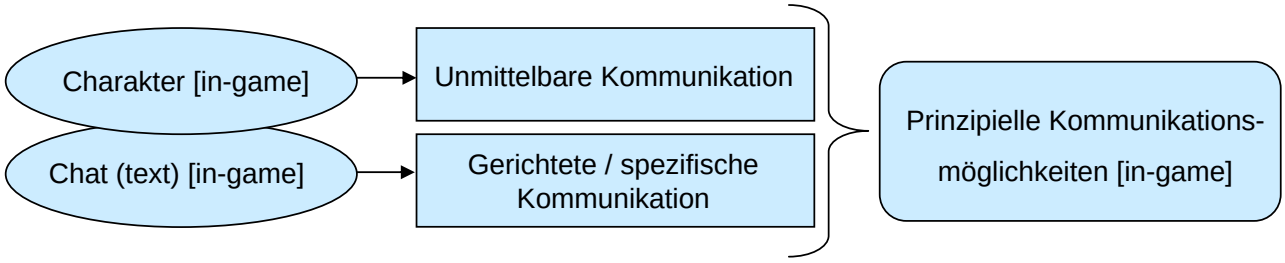

Abbildung 4.4: Prinzipielle Kommunikationsmöglichkeiten innerhalb einer virtuellen Welt

einziges zusätzliches Interface zeitgleich zur virtuellen Welt verwendet wird und dadurch einen direkten Einfluss auf die Interaktionen in der virtuellen Umgebung hat, wird es in dieser Untersuchung ebenfalls betrachtet. Durch Verwendung des Voice-Chats ist eine gerichtete Kommunikation möglich. Anders als in einem Text-Chat (der dem Benutzer die gleichzeitige Verwendung multipler Kanäle ermöglicht) kann sich der Benutzer (mit den heutigen Voice-Chat Tools) immer nur in einem Sprachkommunikationskanal befinden. Durch das zusätzliche out-game Interface ergeben sich, in Ergänzung zu den in-game Kommunikationsmöglichkeiten, noch die *prinzipiellen out-game Kommunikationsmöglichkeiten* (K_{po}, siehe Kommunikationsmodell in Abbildung 4.5).

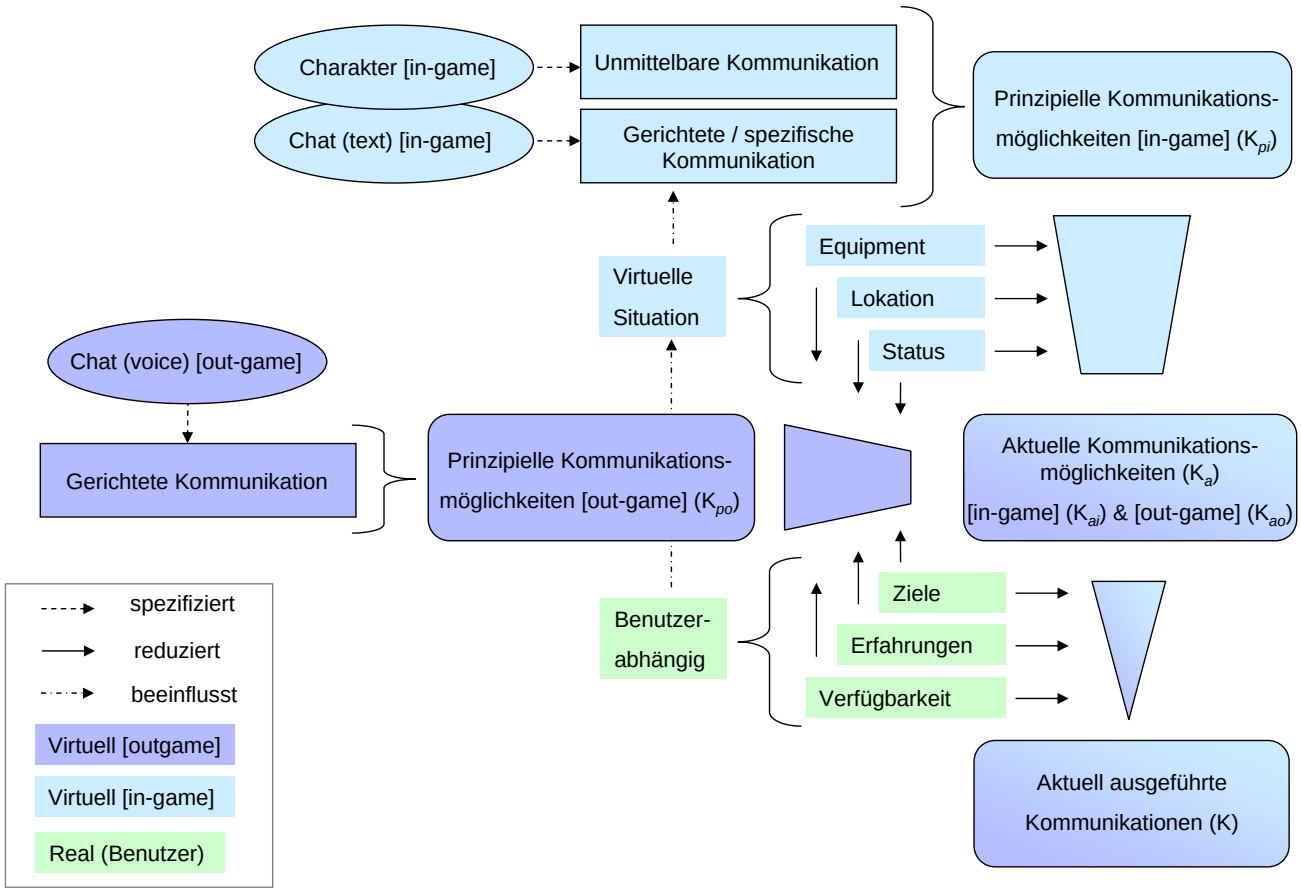

Abbildung 4.5: Modell der Kommunikation

Virtuelle Parameter beeinflussen auch in diesem Fall die in der aktuellen Situation verfügbaren Kommunikationsmöglichkeiten. Für den Fall der in-game Kommunikation besteht zum Beispiel eine Kommunikation zwischen zwei virtuellen Charakteren darin, dass das Gesagte in einer Sprechblase über dem

Kopf des Charakters erscheint. Alle anderen Charaktere, die sich in der Nähe befinden, können nun sehen, dass "etwas" gesagt wurde. Einen lesbaren Text sehen jedoch nur solche Charaktere, die zu der gleichen Fraktion wie der Sprecher gehören. Eventuell muss der virtuelle Charakter in dieser Situation winken anstelle "Hallo" zu sagen, damit der andere Charakter (bzw. sein Benutzer) den Gruß verstehen kann. Somit entsteht abhängig von der virtuellen Situation eine Teilmenge mit *aktuellen in-game Kommunikationsmöglichkeiten* (K_{ai}).

Auch die prinzipiellen out-game Kommunikationsmöglichkeiten werden durch diese virtuellen Parameter beeinflusst. So ist zum Beispiel die aktuelle Sprachkommunikation der virtuellen Gruppe, in der sich der Charakter befindet, genau einem Chatraum zugeordnet, sodass je nach Gruppenzugehörigkeit die Kommunikationsmöglichkeiten unterschiedlich sind. Im Unterschied zu den in-game Kommunikationsmöglichkeiten spielen bei der Reduktion aller prinzipiellen zur Menge der aktuell verfügbaren out-game Kommunikationsmöglichkeiten die realen Parameter schon im ersten Schritt eine Rolle. Verfügt der Benutzer zum Beispiel nicht über das entsprechende externe Voice-Chat Tool oder möchte er es nicht verwenden, so ist in diesem Fall die entsprechende Sprachkommunikation nicht Teil der *aktuellen out-game Kommunikationsmöglichkeiten* (K_{ao}). Im Gegensatz zu den in-game Kommunikationsmöglichkeiten, von denen in jeder Situation zumindest eine Teilmenge verfügbar ist (wie beispielsweise ein Text-Chat), kann die Menge der aktuell verfügbaren out-game Kommunikationsmöglichkeiten leer sein.

Aus der Vereinigung der aktuell verfügbaren (in-game und out-game) Kommunikationsmöglichkeiten ergibt sich die Menge aller *aktuellen Kommunikationsmöglichkeiten* (K_a). Die *aktuell ausgeführten Kommunikationen* (K), die der Benutzer letztendlich wählt, hängen dann wieder von ihm selbst ab. Jene Kommunikationsmöglichkeiten, die am besten zu seinen Vorlieben, Erfahrungen und Zielen passen, wird der Benutzer letztendlich auch verwenden. Im Unterschied zu den Aktionen, die sequenziell ausgeführt werden, kann Kommunikation auch parallel stattfinden (z.B. kann ein Benutzer gleichzeitig im Voice-Chat sprechen und in den Text-Chat schreiben). Zusammengefasst bedeutet das:

- Alle prinzipiell möglichen Kommunikationsmöglichkeiten K_p ergeben sich aus der Summe von in-game und out-game Kommunikationsmöglichkeiten:

  $$K_p = K_{pi} \cup K_{po}$$

  Im ersten Schritt entstehen die Teilmengen K_{ai} und K_{ao} aus den Mengen aller möglichen in-game K_{pi} und out-game K_{po} Kommunikationsmöglichkeiten:

  $$(K_{ai} \subseteq K_{pi}) \wedge (K_{ao} \subseteq K_{po})$$

  Die aktuellen Kommunikationsmöglichkeiten K_a sind somit die Vereinigung der aktuellen in-game und out-game und Teil aller prinzipiellen Kommunikationsmöglichkeiten:

  $$K_a = K_{ai} \cup K_{ao} \wedge K_a \subseteq K_p$$

- Im zweiten Schritt entsteht die Menge K (die in diesem Fall mehrere Elemente k_i enthalten kann, da es möglich ist auf verschiedenen Wegen gleichzeitig zu kommunizieren) aus der Teilmenge K_a:

  $$K \subseteq K_a$$

- Somit ergeben sich die aktuell ausgeführten Kommunikationen:

  $$k_i \in K \wedge K \subseteq K_p \tag{4.2}$$

Zusätzlich zum beschriebenen Ablauf im Kommunikationsmodell gibt es zwei Rückkopplungen, die auch im Aktionsmodell enthalten sind. Auch in diesem Fall ist bei der Modellbetrachtung der Einfluss der virtuellen Parameter auf die Ausgangssituation zu berücksichtigen. Auch der Erwerb neuer Kommunikationsfähigkeiten kann abhängig sein von den bereits erworbenen Fähigkeiten und Erfahrungen und die virtuelle Situation, in die sich ein Benutzer begibt, ist analog auch in diesem Fall benutzerabhängig.

Auch bei der Betrachtung der Kommunikation zeigt sich, dass die Auswahl der Kommunikation, die der Benutzer ausführt, in zwei Schritte unterteilt ist. Trotz der zusätzlichen Kommunikationsmöglichkeiten, die sich durch die verschiedenen Interfaces ergeben, entsteht durch die virtuelle Situation eine Teilmenge mit aktuell verfügbaren Kommunikationsmöglichkeiten aus denen der Benutzter, basierend auf seinen Erfahrungen und Vorlieben, auswählt.

4.1.3 Generisches Interaktionsmodell

Als Schlussfolgerung aus der Analyse der Interaktionsmöglichkeiten in virtuellen Welten wurde ein generisches Modell für Interaktionen in virtuellen Umgebungen entwickelt. Wie sich aus der Analyse von Aktionen in virtuellen Welten ergibt (siehe Abschnitt 4.1.2), erfolgt die Auswahl einer ausgeführten Aktion in zwei Schritten. Aus der Menge der Aktionsmöglichkeiten, die der Benutzer eines virtuelle Charakter prinzipiell zur Verfügung hat, ergibt sich eine situationsabhängige Teilmenge der in der aktuellen virtuellen Situation ausführbaren Aktionsmöglichkeiten. Aus dieser Teilmenge wählt dann der Benutzer die auszuführende Aktion aus. Analog verhält es sich bei der Kommunikation. Auch die Auswahl der ausgeführten Kommunikation erfolgt in zwei Schritten. Die Einflussfaktoren, die in beiden Szenarien eine Rolle spielen, sind identisch (virtuelle und reale Parameter). Daher kann ein verallgemeinertes Modell der Interaktion in virtuellen Welten abgeleitet werden. Der Interaktionstyp der erweiterten Kommunikation ist nicht Bestandteil dieses Modells, da die erweiterte Kommunikation nur außerhalb der virtuellen Welt stattfindet und (anders als die Sprachkommunikation) nicht im direkten Zusammenhang mit der virtuellen Interaktion steht. Das generische Modell der Interaktionsmöglichkeiten fasst die Erkenntnisse der Analysen zusammen und bildet die identifizierten Schritte in Form eines Two-tier Modells (Zweistufenmodell) ab (siehe Abbildung 4.6).

Verallgemeinert ergibt sich im ersten Schritt eine Reduktion der prinzipiell verfügbaren Interaktionsmöglichkeiten (*prinzipielle Interaktionsmöglichkeiten*, I_p) durch die virtuelle Situation. Die virtuellen Parameter bestimmen dabei, ob eine Interaktion in der aktuellen Situation ausgeführt werden kann oder nicht. Die dadurch entstehende Teilmenge an Interaktionsmöglichkeiten (*aktuelle Interaktionsmöglichkeiten*, I_a) enthält die Interaktionen, welche im Moment ausgeführt werden können.

Im zweiten Schritt wählt der Benutzer aus der Menge der Interaktionsmöglichkeiten, die in diesem Moment verfügbar sind, die auszuführenden Interaktionen aus und führt sie aus (*aktuell ausgeführte Interaktionen*, I). Es ist möglich, wie auch bei der Kommunikation, verschiedene Interaktionen gleichzeitig auszuführen (z.B. gleichzeitig im Voice-Chat zu sprechen und mit dem Charakter durch die virtuelle Umgebung zu laufen). Die Auswahl der Interaktion ist abhängig von den realen Parametern des Benutzers.

Das bedeutet:

- Im ersten Schritt entstehen die Menge der aktuellen Interaktionsmöglichkeiten I_a als Teilmenge aller möglichen Interaktionen I_p:

  $$I_a \subseteq I_p$$

- Im zweiten Schritt entsteht die Menge der Interaktionen die ausgeführt werden I (die mehrere Elemente i_i enthalten kann) als Teilmenge der aktuellen Interaktionsmöglichkeiten I_a:

  $$I \subseteq I_a$$

- Somit gilt für die aktuell ausgeführten Interaktionen:

  $$i_i \in I \wedge I \subseteq I_p$$

  Die Menge aller Interaktionsmöglichkeiten I_p setzt sich zusammen aus den prinzipiellen Aktions- und den Kommunikationsmöglichkeiten:

  $$I_p = A_p \cup K_p$$

  Mit den Aussagen (4.1) und (4.2) folgt dann für die ausgeführten Interaktionen:

  $$I = A \cup K \tag{4.3}$$

Die Auswahl der Interaktion durch den Benutzer ist abhängig von seinen realen Parametern. Diese realen Parameter spezifizieren den *realen Kontext* des Benutzers (siehe auch 3.1.1). Analog verhält es sich für die virtuellen Parameter des Charakters, der den Benutzer in der virtuellen Welt repräsentiert. Die virtuellen Parameter beschreiben den *virtuellen Kontext* des virtuellen Charakters.

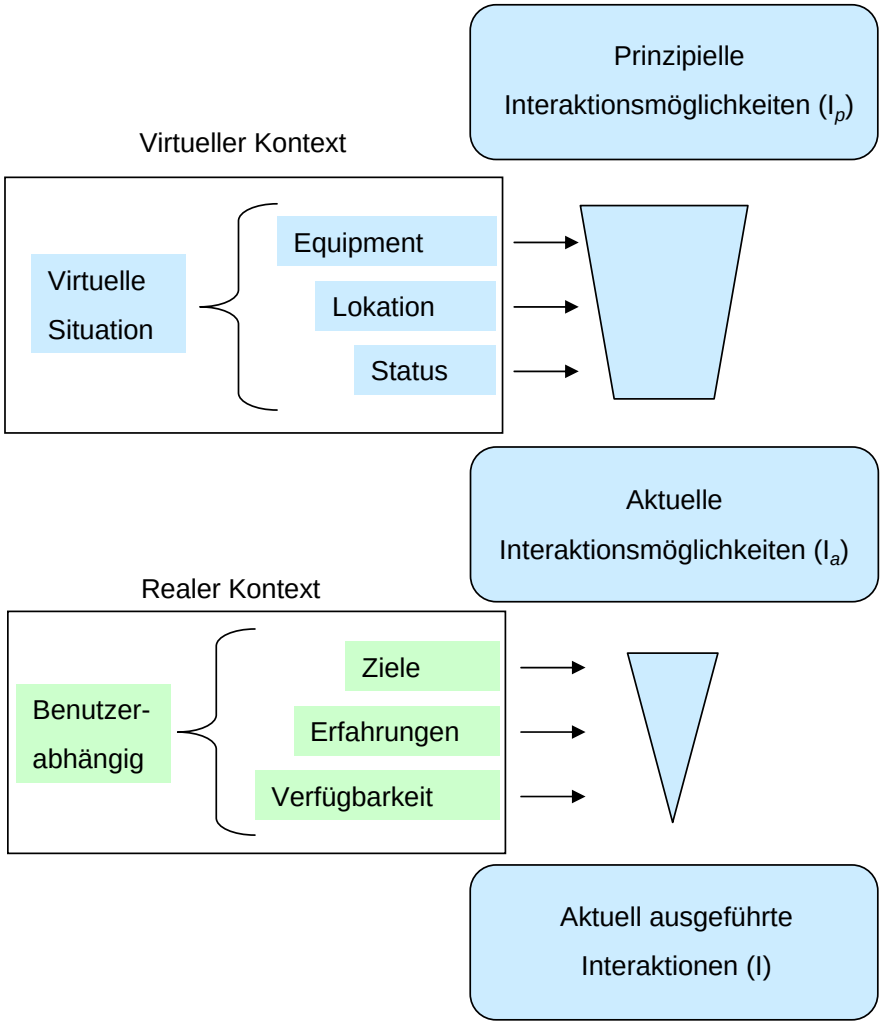

Abbildung 4.6: Generisches Two-tier Modell der Interaktion in virtuellen Welten

Die durchgeführte Analyse umfasst alle grundlegenden Interaktionstypen (Aktion und Kommunikation) und Interfaces, sowie Untersuchungen bezüglich ihrer Übereinstimmungen und Unterschiede. Durch

zusätzliche Interface-Erweiterungen kann die Ausgangsmenge der prinzipiellen Interaktionsmöglichkeiten erweitert werden. Die erweiterte Ausgangsmenge (I'_p) besteht dann aus der Vereinigung der existierenden Ausgangsmenge (I_p) mit der Erweiterungsmenge (I_e): $I'_p = I_p \cup I_e$. Der zweistufige Auswahlprozess und der Einfluss der Kontextparameter bleiben unverändert bestehen.

4.1.4 Bedeutung von Gruppeninteraktion

Kooperation und Koordination von Spielern in MOGs basieren auf Interaktionen, wie Aktion und Kommunikation (vergleiche Kapitel 4.1.2). Die in dieser Arbeit durchgeführte Untersuchung der Interaktion zwischen Spielern hat gezeigt, dass sie in verschiedenen Formen und Ausprägungen in der virtuellen Umgebung der Spiele, zum Beispiel im Rahmen von Handel und Wirtschaft, aber vor allem auch beim gemeinsamen Spielen in der Gruppe stattfindet. Untersuchungen der Interaktion zwischen Spielern in der Literatur (wie zum Beispiel bei [DM04], [NH06] oder [Smi05]) beschränken sich auf die Betrachtung der sozialen Komponente (siehe auch Kapitel 3.3.3). Eine genaue Betrachtung der spielbezogenen Aspekte bleibt ausserhalb der Darstellung. Diese sind aber die Grundlage für die entstehenden Interaktionen in Spielergruppen und auch darüber hinaus. Gute Kommunikation und Koordination ist essenziell für ein Team, um erfolgreich zu sein. Gute Kommunikation und Koordination findet dann statt, wenn die Gruppenmitglieder sich in ihren spezialisierten Rollen aufeinander verlassen können [Che09c]. Beim Gruppenspiel sind die Aufteilung von Aufgaben und die gegenseitige Unterstützung notwendig, um erfolgreich gemeinsame Aufgaben durchführen zu können. Diese Aufgabenteilung findet man bei allen MOGs. Eine besonders anschauliche Gruppenaufteilung bieten Rollenspiele an, weshalb sie im Fokus der folgenden Darstellung der Analyseergebnisse stehen. Dabei sind die entstehenden Gruppeninteraktionen nicht trivial und vor allem für eine Unterstützung der Kooperation von Spielern elementar.

Für das Gruppenspiel in MOGs gibt es zwei grundsätzliche Ausprägungen: Kooperation gegen virtuelle Gegner (*Player versus Environment* - PvE) und gegen andere Spieler (kompetitive oder *Player versus Player* - PvP). Beiden gemein ist, dass jeder Spieler eine Aufgabe oder einen Aufgabenbereich innerhalb seiner Gruppe zugeteilt bekommt. Basierend auf den Fähigkeiten des Charakters und der zugeteilten Aufgabe resultiert die Rolle, die der Spieler im Rahmen der Kooperation zu erfüllen hat. Vor allem für eine Unterstützung der Kooperationen sind die Rollen der Spieler von großer Bedeutung. Eine Grundlage für die Rollenverteilung bilden die Charakterklassen, die eine Spezialisierung des virtuellen Charakters darstellen (siehe 2.1).

In klassenbasierten Spielen ergeben sich die Rollen, die ein Spieler übernehmen kann aus den Fähigkeiten seines virtuellen Charakters. Ein Beispiel sind Charaktere, die hohen Schaden aushalten und deshalb Rollen übernehmen können bei denen sie sich in der direkten Nähe zum Gegner aufhalten müssen, im Gegensatz zu Charakteren, die nur wenig Schaden aushalten und sich dadurch nur für Rollen eignen, die in Entfernung zum Gegner ausgeführt werden können. Somit legt die Charakterklasse bestimmte Rollen nahe und verbietet auch bestimmte Rollen. Trotz des Zusammenhangs zwischen Rolle und Charakterfähigkeiten sollten immer mehrere Klassen für eine Rolle geeignet sein, und jede Klasse sollte mehrere Rollen übernehmen können. Dabei sollte die Anzahl der Rollen, die eine Klasse übernehmen kann, auch beschränkt sein ([San03]). Eine ausgewogene Verteilung von Charakterklassen und Rollen, ist auch wichtig für das *Balancing* eines Spiels, da kein Spieler aufgrund der Wahl seiner Charakterklasse benachteiligt oder bevorzugt sein sollte. Durch den Zusammenhang zwischen Charakterklasse und Rolle werden diese Rollen im weiteren auch als "Charakterrolle" bezeichent.

Neben den Aufgaben, die Charakterrollen zugeordnet werden, gibt es weitere Aufgaben, die unabhängig vom gewählten Charakter ausgeführt werden können. Diese Aufgaben, können charakterunabhängigen "Meta-Rollen" zugeordnet werden. Typische Meta-Rollen sind: *Team Leader* (Gruppenchef) und *Coordinator* (Koordinator).

Team Leader: Der Team Leader ist für die Leitung einer Gruppe verantwortlich. Für diese Aufgabe sind sowohl entsprechende "Soft-Skills" als auch Erfahrungen und Kenntnisse über das Spiel entschei-

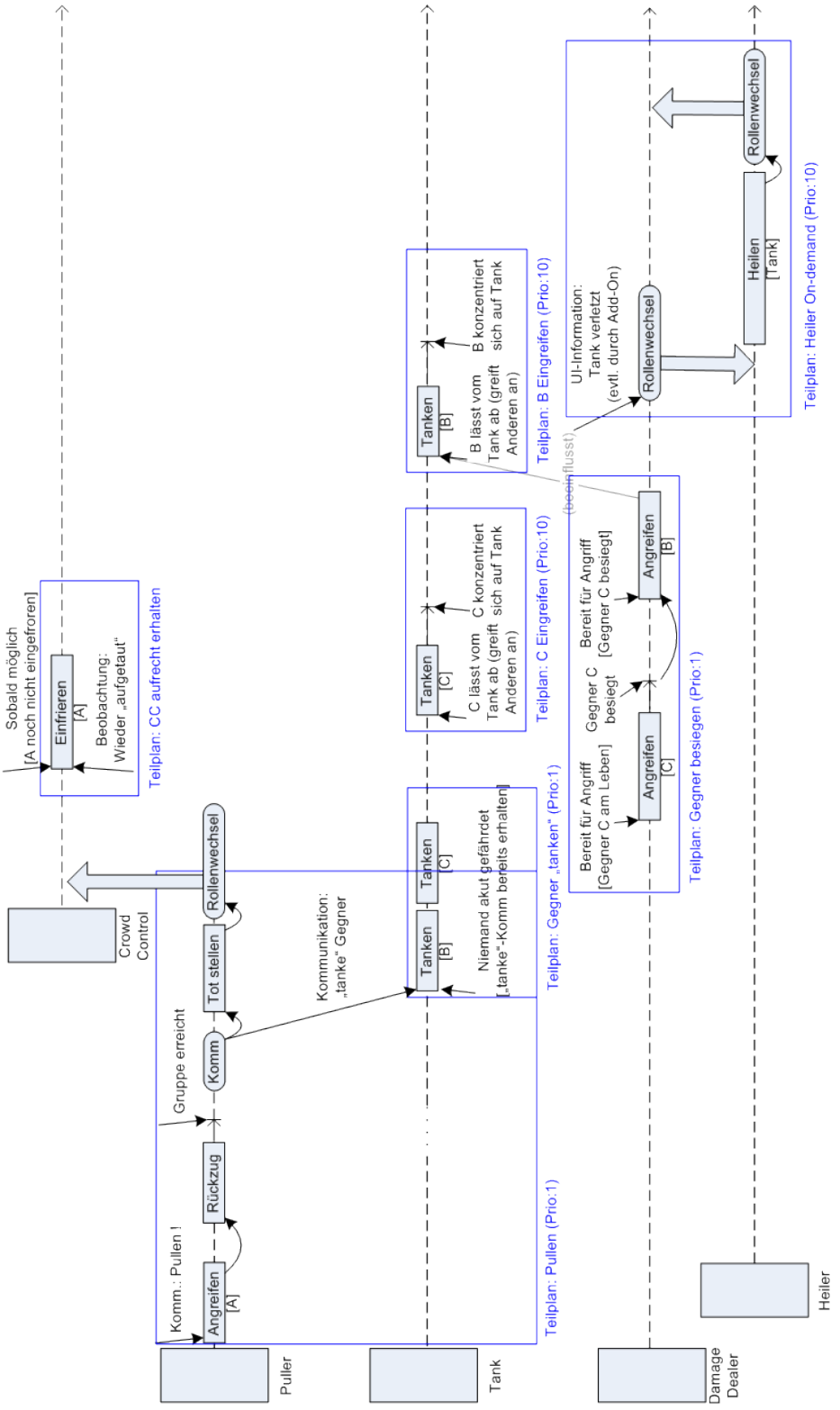

Abbildung 4.7: Beispiel eines Gruppenspiels in einem MMORPG

dend. Der Team Leader verteilt die verschiedenen Aufgaben an die Gruppenmitglieder und initiiert die Aktionen der Gruppe. Wird eine große Gruppe in Untergruppen unterteilt, so benötigen diese wiederum einen Untergruppenleiter, der für die Untergruppe verantwortlich ist, seine Gruppenmitglieder instruiert und den Team Leader über den Zustand seiner Gruppe informiert. Während einer Begegnung koordiniert der Team Leader das Vorgehen und passt die Taktik des Teams dynamisch den Anforderungen an, um auf das Verhalten des Gegners entsprechend reagieren zu können. Der Team Leader hat auch außerhalb der eigentlichen Begegnungen eine wichtige Rolle, da er im Vorfeld die Planung der Vorgehensweise leitet und in der Nachbereitung die Analyse eine Begegnung mit der Gruppe übernimmt.

Koordinator: Für diese Rolle existieren eine Reihe unterschiedlicher Bezeichnungen wie Main Assist, Caller, Timer, Reporter, etc.. Die Aufgabe des Koordinators ist es, den Team Leader zu unterstützen, indem er spezielle Aufgaben übernimmt, die für die Koordination der Gruppe oder den Ablauf einer Begegnung wichtig sind. Solche Aufgaben können sein: aus den Gegnern ein Ziel auszuwählen, auf das sich die Gruppe konzentrieren soll, die Aktionen der Gegner zu überwachen und darüber zu berichten, bestimmte Events anzusagen, im Spiel vorhandene Zusatzaktionen zu verfolgen oder Spezialaufgaben durchzuführen. Für die Rolle des Koordinators sind vor allem Erfahrung und Kenntnisse über die Spielsituation entscheidend. Für manche Spezialaufgaben können bestimmte Fähigkeiten von Vorteil sein.

Wichtig für die Kooperation in der Gruppe ist, dass ein Spieler mehrere Rollen erfüllen kann und dadurch dynamisch, je nach Situation, seine Rolle wechselt. Abbildung 4.7 zeigt ein Diagramm, das einen typischen Ablauf einer Gruppenspielsituation modelliert. Diese Darstellungsform wurde im Rahmen dieser Arbeit entwickelt, um Interaktionen in MOGs modellieren zu können. Dargestellt wird ein Auszug aus einer Begegnung zwischen drei Spielern und drei Gegnern (A, B, C). Die drei Spieler erfüllen in diesem Ausschnitt fünf verschiedene Rollen, ein Spieler über nimmt die Rolle des *Tank*, ein weiterer Spieler ist *Damage Dealer* und *Supporter* (*Heiler*) und der dritte Spieler ist *Supporter* (*Crowd Control*) und hat eine spezielle *Coodinator*-Rolle (*Puller*). Die Form der Darstellung ist angelehnt an UML-Sequenzdiagramme und wird im Anhang beschrieben (siehe A.3).

4.2 Virtueller Kontext

Basierend auf den durchgeführten Analysen und Untersuchungen kann nun die Frage danach, was virtueller Kontext ist, beantwortet werden. Dazu folgt zunächst die auf den Untersuchungsergebnissen basierende Definition von virtuellem Kontext sowie im Anschluss die Darstellung der virtuellen Parameter und ihrer Eigenschaften.

4.2.1 Definition von virtuellem Kontext

Im Gegensatz zu Kontext in der realen Welt "realem Kontext", der sich auf die reale Situation des Benutzers bezieht (wie z.B. der Ort an dem sich der Benutzer aufhält), bezieht sich virtueller Kontext auf die virtuelle Repräsentation des Benutzers und dessen Situation in der virtuellen Umgebung.

In Anbetracht der Definitionen von Kontext in der realen Welt (vergleiche 3.1.1) und basierend auf der Analyse von virtuellen Welten wird folgende Definition getroffen:

Definition 4.1 ***Virtueller Kontext*** *beschreibt die Beziehung zwischen virtuellen Objekten (inklusive virtuellen Charakteren) und der virtuellen Umgebung (virtuellen Welt).*

So wie reale Kontextinformationen in vier Kontexttypen (Lokation, Identität, Zeit und Aktivität, siehe Abschnitt 3.1.1) eingeteilt werden, können auch virtuelle Kontextinformationen in vier Kategorien unterteilt werden. Die Kategorien sind an die vier realen Kontexttypen angelehnt. Im Unterschied zu den realen Kontexttypen sind die virtuellen Parameterkategorien jedoch etwas weiter gefasst. Dadurch können alle virtuellen Parameter diesen Parameterkategorien zugeordnet werden. Die vier Kategorien sind

Generelle Informationen, die Angaben über die Zeit und andere allgemeine Informationen enthalten, **Lokationsinformationen**, die Informationen darüber, wo sich der virtuelle Charakter in der virtuellen Umgebung aufhält, enthalten, **Charakterinformationen**, die die Identität des virtuellen Charakters beschreiben und **Statusinformationen**, die die Aktionen des Charakters und die Beziehungen zu anderen virtuellen Charakteren beinhalten (siehe auch Abschnitt 4.3.2).

Virtuelle Welten sind durch ein Regelwerk spezifiziert, welches die virtuelle Welt beschreibt. Das bedeutet, dass die Menge der möglichen Interaktionen eines virtuellen Charakters beschränkt ist. Virtuelle Parameter können, dadurch dass die Anzahl der virtuellen Kontextinformationen durch die virtuelle Welt vorgegeben und somit beschränkt ist, die virtuelle Situation eines virtuellen Charakters in einer virtuellen Welt genau spezifizieren. Alle Kontextinformationen, die es in der virtuellen Welt gibt, können erfasst und verwendet werden. Dies ist in der realen Welt aufgrund der unbegrenzten Menge an Kontextinformationen so nicht möglich.

4.2.2 Virtuelle Parameter

Basierend auf der Analyse der Interaktionsmöglichkeiten in virtuellen Welten wurden *reale Parameter* und *virtuelle Parameter* identifiziert (siehe auch 4.1.2). Die realen Parameter beschreiben bestimmte Aspekte des Benutzers einer virtuellen Welt. Die virtuellen Parameter charakterisieren die virtuelle Situationen der virtuellen Repräsentation des Benutzers.

Die **realen Parameter** beschreiben die Aspekte des Benutzers, die einen Einfluss darauf haben, welche Interaktion in der virtuellen Welt er ausführt. Beispiele von realen Parametern sind: Verfügbarkeit (ist der Benutzer nicht an der Tastatur (afk), wie schnell ist seine Internetverbindung, wieviel Zeit verbringt er in der virtuellen Welt), Erfahrungen (welche Spielerfahrungen hat der Benutzers, was hat der Benutzer in der virtuellen Welt schon alles getan, wie informiert er sich zusätzlich, wie gut kennt er sich aus), Ziele (was möchte der Benutzer tun, welche Ziele möchte er erreichen) und Vorlieben (was bevorzugt der Benutzer, was möchte er auf keinen Fall).

Reale Parameter sind Informationen, die den Kontext des Benutzers charakterisieren. Die realen Parameter entsprechen somit den in Abschnitt 3.1.1 eingeführten Kontextinformationen. Da es sich um Kontextinformationen des realen Benutzers handelt beschreiben diese den realen Kontext des Benutzers.

Die **virtuellen Parameter** spiegeln die virtuelle Situation des virtuellen Charakters eines Benutzers wieder. Es sind Informationen, die den Kontext des virtuellen Charakters charakterisieren. Virtuelle Parameter sind virtuelle Kontextinformationen. Einige dieser Parameter sind abhängig von den Fertigkeiten des Charakters. Beispiele für virtuelle Parameter, die üblicherweise in einer virtuellen Welt vorkommen sind Lokation, Status und Equipment. Die Lokation beinhaltet Informationen über die Position des virtuellen Charakter innerhalb der virtuellen Welt wie Koordinaten, Umgebung oder sein Sichtfeld (FOV - *field of view*). Der Status umfasst Informationen über den Zustand eines virtuellen Charakters (Lebenspunkte, aktuelle Aktion, ...) und seine Beziehungen zu anderen Charakteren (Gruppe, Gilde, ...). Equipment bezeichnet Informationen über die Ausrüstung des virtuellen Charakters (Kleidung, Gegenstände im seinem Besitz, Schmuck, Inhalt seiner Taschen, ...).

Im Unterschied zu den realen Parametern handelt es sich bei den virtuellen Parametern um Kontextinformationen einer virtuellen Repräsentation des Benutzers und nicht des realen Benutzers selbst. Kontextinformationen des virtuellen Charakters beschreiben seinen Kontext in Bezug auf die virtuelle Welt (*virtueller Kontext*).

Neben den virtuellen Parametern, die den Kontext des virtuellen Charakters beschreiben, hat ein virtueller Charakter *virtuelle Eigenschaften* (Befugnisse und Funktionen). Diese Befugnisse und Funktionen eines Charakter sind abhängig von seinen Fähigkeiten und von seiner Historie (das was er getan hat) in der virtuellen Welt (siehe Beispiel des schnellen Pferdes in 4.1.2).

Die virtuellen Eigenschaften eines Charakters sind wichtig für die Interaktion mit anderen Benutzern (z.B. ist die Rolle eines Benutzers innerhalb einer Gruppe abhängig davon) und können die virtuellen Parameter eines Charakters beeinflussen.

4.2.3 Eigenschaften virtueller Parameter

Eine detaillierte Betrachtung der Eigenschaften virtueller Parameter bildet die Grundlage für die Spezifikation einer geeigneten Beschreibung für virtuellen Kontext. Die Betrachtung der Eigenschaften virtueller Parameter ist Teil der durchgeführten Analyse virtueller Welten. Die entsprechenden Ergebnisse werden im Folgenden vorgestellt. Die durchgeführte Analyse virtueller Welten hat gezeigt, dass virtuelle Parameter unterschiedliche Eigenschaften haben und sich beispielsweise in ihrem Typ, in ihrer Komplexität oder darin unterscheiden, wie schnell und oft sich ihre Werte ändern.

Auch reale Kontextinformationen besitzen bestimmte Eigenschaften. Dazu zählen der Bezug zu Zeit und Lokation sowie Gültigkeit, Unsicherheit und Unschärfe des an den Kontextparameter gebundenen Wertes [Mos02]. Die Lokation spielt gerade für realen Kontext eine wichtige Rolle, da in der echten Welt alle Angaben (sogar die Angabe der Zeit) je nach Lokation (Zeitzonen) unterschiedlich sein kann. In der virtuellen Welt gibt es diese Unterschiede nicht, da solche grundsätzlichen Festlegungen sich immer "global" auf die virtuellen Welt beziehen. Die Zeit jedoch nimmt auch bei virtuellem Kontext eine besondere Rolle ein, da die Werte der Kontextparameter immer zu einem bestimmten Zeitpunkt gültig sind. Aus diesem Grund werden alle virtuellen Kontextinformationen mit einem Zeitstempel (*timestamp*) versehen. Die weiteren Eigenschaften der realen Kontextinformationen lassen sich nicht auf virtuelle Kontextinformationen übertragen, da die realen Kontextparameter mittels Sensoren erfasst werden und dabei weitere Effekte wie Messungenauigkeiten betrachtet werden müssen.

Es gibt in virtuellen Welten bestimmte Eigenschaften virtueller Parameter (die, abstrakt betrachtet, das virtuelle Äquivalent der Sensordaten darstellen), die bei der Modellierung und Beschreibung der Parameter berücksichtigt werden müssen. Diese Eigenschaften virtueller Parameter sind: Wertebereich, Komplexität, Variabilität und Änderungsrate.

- **Wertebereich**: Der Wertebereich gibt an, in welchem Bereich die Werte, die ein virtueller Parameter annehmen kann, liegen und welchen Typ diese haben. Die Werte eines Parameters können verschiedene Typen haben (z.B. boolesch, nummerisch oder Text), die Werte können alle je Parameter auf einen Typ beschränkt sein oder auch kombiniert vorkommen (siehe Komplexität).

- **Komplexität**: Die Komplexität eines virtuellen Parameters beschreibt die Struktur und den Umfang der Information (Anzahl der Werte), die ein virtueller Parameter enthält. Struktur und Umfang eines virtuellen Parameters können durch eine Datenstruktur abgebildet werden. Diese kann entweder einfach oder komplex sein. Eine einfache Datenstruktur kann aus nur einem Wert bestehen wie es zum Beispiel bei dem virtuellen Parameter *Hitpoints* (Lebenspunkte) der Fall ist, eine komplexe Struktur kann mehrere geschachtelte Ebenen mit gemischten Werten enthalten zum Beispiel der Parameter *Handwerk*, der die verschiedenen *Berufe* sowie jeweils die Güte der handwerklichen Fähigkeiten des Charakters (*Skill Level*) und die jeweiligen *Rezepte*, die der Charakter herstellen kann, enthält.

- **Variabilität**: Über die Variabilität der Werte kann beschrieben werden, wie stark sich ein Wert von seinem Vorgänger unterscheiden kann. Variabilität $V = \Delta(Vt_1, Vt_2)$ wobei Vt_1 der Parameterwert zum Zeitpunkt t_1 und Vt_2 der Parameterwert zum Zeitpunkt t_2 ist. Eine geringe Variabilität hat zum Beispiel der virtuelle Parameter *Level*, dieser kann sich nur in eine Richtung ändern und das auch nur inkrementell. Eine hohe Variabilität hingegen hat zum Beispiel der Parameter *Equipment*, so kann ein virtueller Charakter etwa den Gegenstand, den er in der Hand hält, von einem Schwert direkt zu einer Angel wechseln.

- **Änderungsrate**: Die Änderungsrate eines Parameters ist die durchschnittliche Häufigkeit, mit der sich der Parameterwert ändert. Es gibt Parameter, deren Wert sich selten ändert (*niedrige Änderungsrate*), wie zum Beispiel beim virtuellen Parameter Gilde Eine Gilde ist eine feste Organisationsstruktur, die in der Regel über einen längeren Zeitraum konstant bleibt. Der Wert andere Parameter ändert sich häufig (*hohe Änderungsrate*), wie zum Beispiel die Positions-*Koordinaten*.

Die Eigenschaften der virtuellen Parameter sind vor allem zur optimalen Gestaltung des Austauschs und der Übertragung virtueller Parameter relevant, da dadurch Anforderungen zum Beispiel bezüglich der Updaterate entstehen. Variabilität und Änderungsrate bestimmen die Bedeutung einzelner Parameterwerte bei deren Übertragung. Wenn beispielsweise die Übertragung eines Parameterwertes fehlschlägt, dann ist der Informationsverlust bei Parametern mit niedriger Variabilität eher gering, vor allem wenn der Parameter zugleich eine hohe Änderungsrate besitzt. Bei hoher Variabilität ist auch der Informationsverlust hoch, vor allem bei niedriger Änderungsrate. In diesem Fall muss ein verlorener Wert noch einmal übertragen werden. Bei geringer Variabilität kann davon ausgegangen werden, dass, auch bei Parametern mit einer hohen Änderungsrate, zwei aufeinanderfolgende Werte einen gewissen Unterschied nicht überschreiten. In diesem Fall kann eine Übertragungsform gewählt werden, die schnell ist, aber dafür auf Absicherungsmechanismen (z.B. *Acknoledgements*) verzichtet. Für eine weiterführende Betrachtung wird auf Kapitel 6 verwiesen.

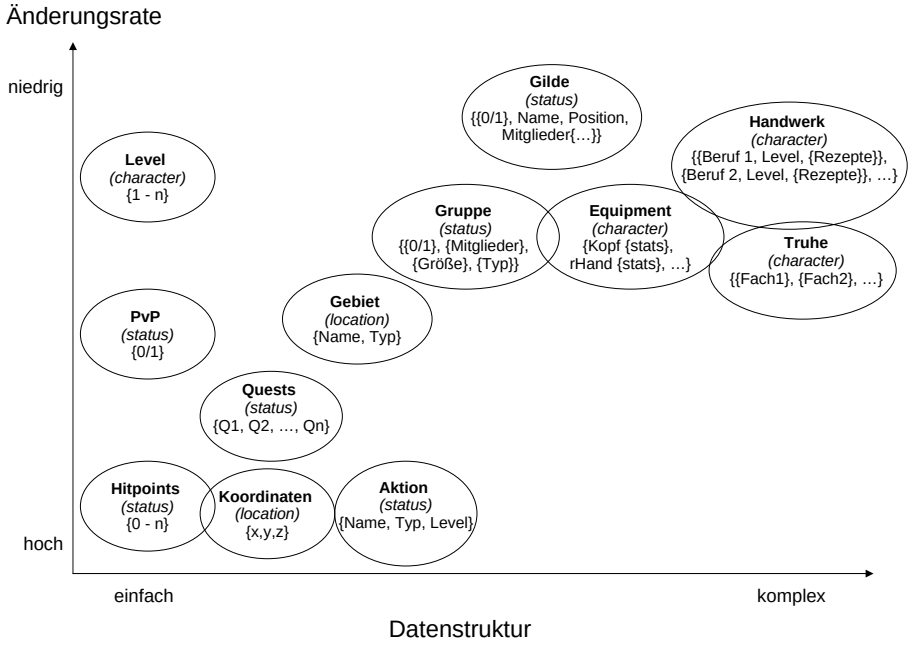

Abbildung 4.8: Eigenschaften virtueller Parameter anhand typischer Beispiele aus MMORPGs

Abbildung 4.8 zeigt exemplarisch, welche Eigenschaften typische virtuelle Parameter in der Regel haben. Die virtuellen Parameter in Abbildung 4.8 sind Beispiele von virtuellen Parametern, die in MMORPGs vorkommen, mit ihren typischen Eigenschaften. Auf der x-Achse ist die *Komplexität* der Datenstruktur aufgetragen und auf der y-Achse die *Änderungsrate* der Parameterwerte. Die Parameter enthalten eine abstrahierte Beschreibung der Parameterwerte, welche die Komplexität darstellt und zulässige Werte (Wertebereich und Variabilität) beispielhaft darstellt. Im Folgenden wird die abstrahierte Beschreibung der Parameter dargestellt. Für den virtuellen Parameter der die Ausrüstung beschreibt, ist das beispielsweise: (1) der Name des Parameters (*Equipment*), (2) die Kategorie des Parameters und (3) eine abstrahierte Darstellung der Datenstruktur (die in diesem Beispiel aus ungefähr 15 Stellen am Charakter wie dem Kopf und der rechten Hand mit den entsprechenden Gegenständen (z.B. Helm, Schwert, Ring) und ihren Werten (*stats*) besteht. Im Unterschied dazu ist eine Datenstruktur des virtuellen Parameters, der die Lebenspunkte des virtuellen Charakters beschreibt, weniger komplex, seine abstrahierte Beschrei-

bung ist beispielsweise: (1) der Name des Parameters (*Hitpoints*), (2) die Kategorie des Parameters und (3) ein abstrahierte Darstellung der Datenstruktur, die einen ganzzahligen Wert (zwischen 0 und der Maximal-Gesundheit des Charakters) enthält.

In der Abbildung lassen sich zwei Cluster erkennen, die eine Häufung von Parametern enthalten. Das sind zum einen die sich häufig ändernden Parameter mit einfacher Struktur in der linken unteren Ecke und zum andern die sich seltener ändernden Parameter mit komplexer Struktur in der oberen rechten Ecke. Das liegt daran, dass Korrelationen zwischen den verschiedenen Parametereigenschaften vorhanden sind. So zeigt sich, dass Werte mit einer hohen Komplexität sowohl eine geringe Änderungsrate als auch eine hohe Variabilität haben, oder dass Werte mit einer hohen Änderungsrate eine geringe Komplexität und eine eher geringe Variabilität aufweisen. Analog zu den Parametern in diesem Beispiel, verhält es sich mit virtuellen Parametern, die in anderen Genres vorkommen wie FPS (First Person Shooter), RTS (RealTime Strategy Game) oder Racer. Das bedeutet, dass Komplexität, Änderungsrate und Variabilität über verschiedene Genres hinweg verallgemeinert werden können, auch wenn die Interpretation der Parameter unterschiedlich ausfallen kann.

Virtuelle Parameter werden durch ein Spiel instantiiert und entsprechend interpretiert. Diese spielspezifische Interpretation kann für verschiedene MOGs sehr ähnlich ausfallen (z.B. der Charaktername), aber sie kann auch ganz unterschiedlich ausfallen. Ein anschauliches Beispiel für eine abweichende Interpretation ist die Betrachtung des Equipments im Fall eines RPGs (RolePlay Games) und eines RTS. In einem RPG besitzt ein virtueller Character bestimmte Kleidungsstücke und Accessoires mit denen er ausgerüstet ist. Das Equipment in einem RTS beinhaltet beispielsweise die Gebäude, die unter der Kontrolle des Charakters stehen.

a)

```
Data Example_RTS {
    player_name              # name of the player
    opponent_count           # number of opponents
    map_name                 # name of the map (map ID)
    coordinates              # location of players base
    ressources {             # ressources the player posseses
        resource_name        # name of the resource
        resource_quantity    # number of the resource
    }
    units {                  # units the player posesses
        unit_name            # name of the unit
        unit_quantity        # number of the unit
        unit_quality         # level of the unit
    }
    buildings {              # buildings the player posseses
        building_quantity    # number of the bulding
        building_quality     # level of the bulding
    }
}
```

b)

```
Data Example_RPG_Character {
    character_name           # name of the character
    character_class          # class of the character
    character_level          # level of the character
    equipment {              # equipment of the character
        head                 # characters helmet
        breast               # characters amor
        weapon               # characters weapon
        bag {                # content of characters bag
            item_name        # name of item contained in bag
            item_count       # number of this items
        }
    }
    skills {                 # skills of the character
        skill_name           # name of the skill
        skill_level          # level of the skill
    }
}
```

c)

```
Data Example_FPS {
    player_name              # name of the player
    map_name                 # name of map
    map_time                 # remaining time for this map
    weapons {
        weapon_name          # name of the weapon
        ammo_count           # munition left for this weapon
    }
    deaths                   # how many times has the player been killed
    kills                    # how many opponets have been killed by the player
    hits                     # how many times did the player hit an opponent
    misses                   # how many times did the player miss an oponent
}
```

Abbildung 4.9: Beispiele für virtuelle Parameter aus verschiedenen MOGs: a) RealTime Strategy Game (RTS), b) RolePlay Game (RPG), c) First Person Shooter (FPS)

Betrachtet man gängige Parameter verschiedener Typen von MOGs, so stellt man fest, dass sich viele Parameter vermeintlich stark voneinander unterscheiden (siehe Abbildung 4.9 mit Beispielparametern aus RTS, RPG und FPS). Diese Beispiele stellen typische Parameter dar, die gewöhnlich durch das Benutzerinterface des MOGs für den Spieler expliziert werden.

Es zeigt sich jedoch, dass es Parametertypen gibt, die in allen Typen von Spielen zu finden sind, da sie grundlegende Kontextangaben enthalten, die unabhängig von einer konkreten Ausprägung einer virtuellen Welt immer von Bedeutung sind (wie zum Beispiel die Lokation in der virtuellen Umgebung). Abbildung 4.10 zeigt virtuelle Parameter, die in allen Spielen vorkommen und konkrete Beispiele dieser Parameter aus verschiedenen MOG Genres. Die Kontextparameter können in verschiedene Klassen unterteilt werden (siehe auch 4.2.1).

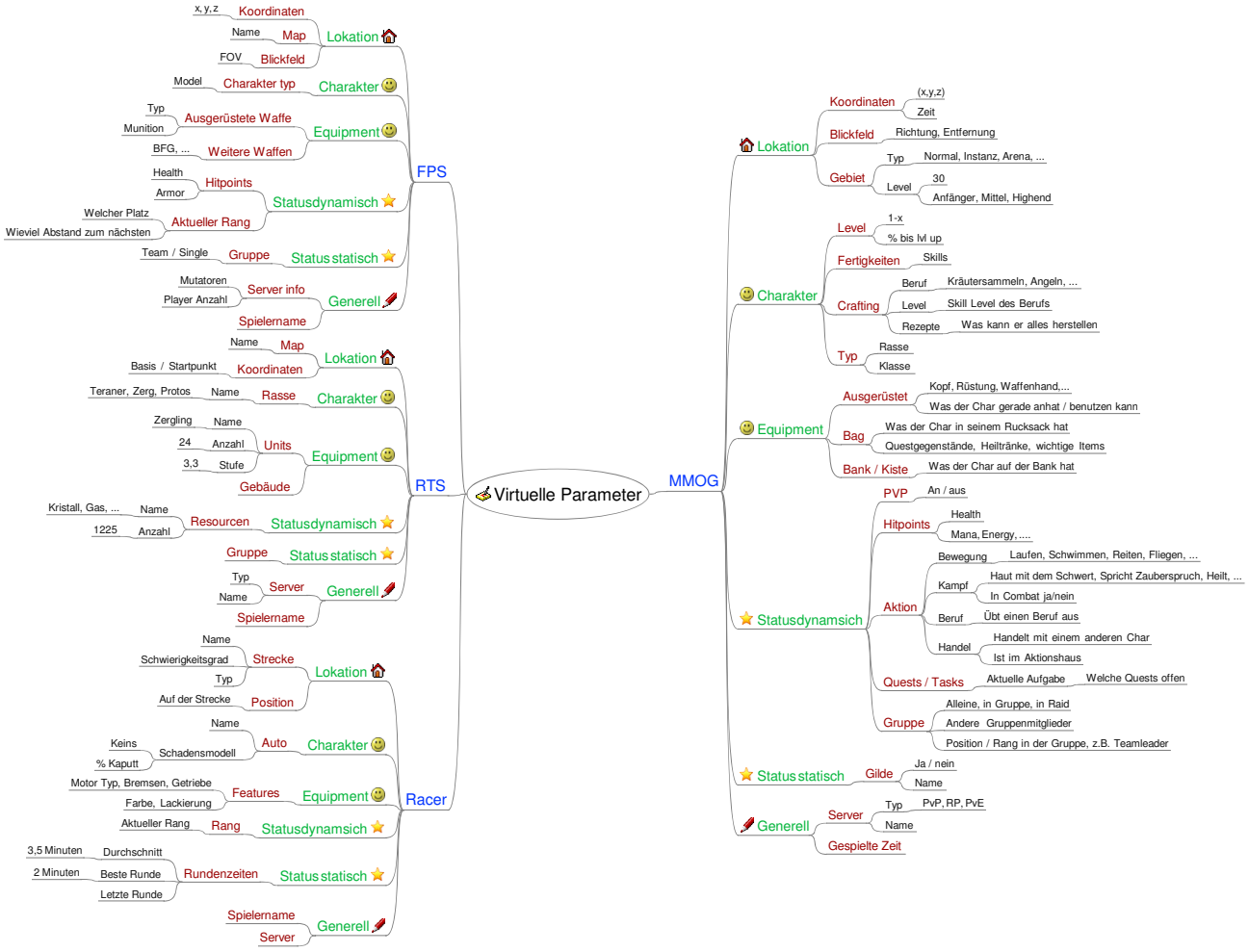

Abbildung 4.10: Einteilung von Parameter-Beispielen aus verschiedenen MOGs

Diese "grundlegenden Kontextparameter" sind ein Subset der virtuellen Parameter und können für alle MOGs generalisiert werden. Für die Kategorisierung der Parameter ergeben sich vier Parameterklassen, in die die virtuellen Parameter eingeordnet werden können:

- **Lokationsinformationen** ("*Lokation*"): Positionsinformationen (z.B. die Koordinaten des Charakters in der virtuellen Welt oder Gebietsinformationen). Die Lokationsinformationen haben einen besonderen Stellenwert, wie auch beim realen Kontext ist die Position des Benutzers beziehungsweise seines Charakters fast immer von Bedeutung. Aus diesem Grund wird die Lokation als eigenständige Kategorie betrachtet.

- **Charakterinformationen** ("*Charakter*" und "*Equipment*"): Angaben über den virtuellen Charakter und seiner Eigenschaften (z.B. Charaktername) sowie sein Equipment. Da sich beide Typen direkt auf die Eigenschaften des virtuellen Charakter beziehen, werden sie in einer Kategorie zusammengefasst.

- **Statusinformationen** ("*Status dynamisch*" und "*Status statisch*"): Informationen über den aktuellen Zustand des Charakters (z.B. die aktuell ausgeführte Aktion) Gruppeninformationen (z.B. ob der Charakter zu einer Gruppe gehört) und andere Zustandsinformationen. Die zwei Kategorien werden zusammengefasst, da beide den Status des virtuellen Charakters betrachten. Darüber hinaus ist der Status auch nur bezogen auf eine bestimmte Zeitspanne als statisch zu betrachten. Diese Zeitspanne variiert jedoch zwischen den verschiedenen Genres, sodass es keine gemeinsame feste Grenze gibt.

- **Generelle Informationen** ("*Generell*"): Allgemeine Informationen wie der Spielername, Zeitangaben (z.B. die gesamte Zeit, in der sich der Spieler in dieser virtuellen Welt schon aufgehalten hat) und Serverinformationen (z.B. Servername). Die generellen Informationen sind im Unterschied zu den anderen virtuellen Kontextinformationen Informationen, die die virtuelle Welt betreffen, in der sich der Benutzer befindet.

4.3 Beschreibung von virtuellem Kontext

Für die Verwendung von virtuellem Kontext müssen die virtuellen Kontextinformationen zunächst erfasst und gespeichert werden. Die größte Herausforderung bei der Erfassung und Beschreibung von virtuellem Kontext ist es, eine Abstraktion der Kontextinformationen zu finden. Anders als bei realem Kontext gibt es nicht nur eine Grundlage (eine einzige Welt) sondern von virtueller Welt zu virtueller Welt unterschiedliche Grundlagen und somit auch unterschiedliche Kontextinformationen. Ein Beispiel ist die Temperatur, diese Kontextinformation ist in der realen Welt immer verfügbar und kann über einen geeigneten Sensor erfasst werden. In virtuellen Welten gibt es jedoch nicht zwangsläufig Wetter und somit auch keine zugehörigen Kontextinformationen, wie die Temperatur, die das Wetter beschreiben. Eine der grundlegenden Herausforderungen ist es somit, eine Beschreibung zu finden, die zum einen mächtig genug ist, virtuelle Situationen zu beschreiben, die aber gleichzeitig verallgemeinert werden kann, flexibel und erweiterbar ist.

Der virtuelle Kontext, beziehungsweise die virtuellen Kontextinformationen sind in virtuellen Welten durch die virtuellen Parameter implizit enthalten. Stellt die virtuelle Welt bzw. ihre Client-Applikation eine Schnittstelle zur Verfügung, können die Kontextinformationen in Form der virtuellen Parametern abgefragt, d.h erhoben werden. Da die virtuellen Parameter abhängig vom Regelwerk einer virtuellen Welt definiert sind, sind sie für jede virtuelle Welt spezifisch definiert. Um eine Kontextbeschreibung auf Basis virtueller Parameter zu entwickeln, ist eine Generalisierung und Kategorisierung der virtuellen Parameter notwendig. Im Folgenden wird die in dieser Arbeit entwickelte XML-basierte Beschreibung von virtuellen Kontext eingeführt und die vorgenommene Kategorisierung und Generalisierung der virtuellen Parameter wird vorgestellt.

4.3.1 Eine Sprache zur Beschreibung von virtuellem Kontext

Für die Erfassung, Speicherung und Verwendung von virtuellen Kontextinformationen wird ein einheitliches Format benötigt, welches eine Beschreibung von virtuellem Kontext ermöglicht. Nach Moschgath [Mos02] haben sich XML [70] und RDF [71] als geeignet für die Beschreibung von realem Kontext herausgestellt. Sowohl XML als auch RDF erfüllen die Anforderungen an eine Spezifikationssprache für Kontextinformationen. Bis auf die Beschreibung von Beziehungen zwischen Kontextinformationen ist jedoch XML besser geeignet.

Diese Erkenntnisse von Moschgath lassen sich auf virtuelle Kontextinformationen übertragen. Das in der Arbeit betrachtet Szenario stützt die Verwendung einer XML-basierten Auszeichnungssprache für die Beschreibung des virtuellen Kontexts. Die Auswahl von XML statt RDF wird untermauert durch die weite Verbreitung von XML in Internet-Applikationen. Darüber hinaus ist XML lizenzfrei, plattformunabhängig und gut unterstützt.

Eine weitere Grundlage für die Spezifikation einer Beschreibungssprache für virtuellen Kontext sind die grundlegenden Kontextparameter (virtuelle Parameter) und ihre Kategorisierung. Die Definition der entwickelten Sprache für die Beschreibung von virtuellem Kontext "xgdl" (e**X**tensible **G**ame **D**escription **L**anguage) basiert auf den Ergebnissen der Analyse der Eigenschaften virtueller Parameter.

XML

Die e**X**tensible **M**arkup **L**anguage (**XML**) [13] wurde ursprünglich für das elektronische Publizieren im großen Umfang entwickelt und spielt inzwischen eine immer wichtigere Rolle für den Austausch verschiedenster Daten. Die Entwicklung von XML begann 1996 und seit 1998 ist es ein W3C-Standard (World Wide Web Consortium) [72]. XML ist ein Nachkomme von SGML (Standard Generalized Markup Language). Die Vorteile von XML sind vor allem die Erweiterbarkeit wie auch die Trennung von Inhalt und Formatierung. Die Beschreibung von XML Dokumenten erfolgt über Schemata, die Regeln für den Aufbau und Inhalt der Dokumente dieses Typs beinhalten. XML ist darüber hinaus selbst eine Definitionssprache für Auszeichnungssprachen, da es Konzepte und Regeln bereitstellt welche die Definition eigener Auszeichnungssprachen ermöglichen. Auszeichnungssprachen, die mit XML definiert werden sind (wie HTML) logisch / semantische Sprachen [MN05]. Dabei bietet XML das Regelwerk (oder Schema) für die Definition neuer Auszeichnungssprachen wobei die unterschiedlichen XML-basierten Sprachen über Namensräume (*namespaces*) eindeutig definiert sind und somit kombinierbar und wiederverwendbar werden.

Speziell für die Definition von Schemata für XML Dokumente wurden Document Type Definition (DTD) [69] und XML Schema [73] entwickelt. **XML DTD** ist Teil der XML Spezifikation [13] und enthält grundlegende Funktionen für die Beschreibung von Schemata. Eine DTD enthält oder referenziert Markup Deklarationen, die eine Grammatik für eine Klasse von Dokumenten bereitstellen. **XML Schema** [FW04] ist eine speziell entwickelte Schema-Sprache. XML Schemata erlauben die Definition von Strukturen, Inhalten und Semantik von XML Dokumenten. Eine mit XML Schema definierte Beschreibung wird als XML-Schema-Definition (XSD) bezeichnet.

Die Definition der Syntax einer Sprache, die auf XML basiert kann über eine DTD oder mit XML Schema erfolgen, wobei XML Schemata einige Vorteile gegenüber DTDs bieten und darum XML Schema heute als Nachfolger der DTD betrachtet wird: höhere Flexibilität und Erweiterbarkeit, Unterstützung von primitiven und komplexen Datentypen und Namensräumen, robusterer Datenaustausch und darüber hinaus werden XML Schemata wiederum in XML geschrieben. Ein XML Schema definiert [75]:

- Elemente, die in einem Dokument vorkommen können,
- Attribute, die in einem Dokument vorkommen können,
- ob Elemente Kinder anderer Elemente sind,
- die Reihenfolge der Kind-Elemente,
- die Anzahl der Kind-Elemente,
- ob ein Element leer ist oder Text beinhalten kann,
- Datentypen für Elemente und Attribute,
- Standardwerte (*default values*) und feste Werte für Elemente und Attribute.

Dabei beinhalten Schema-Komponenten das abstrakte Datenmodell des Schemas. Ein XML Schema besteht aus Komponenten, die in drei Gruppen unterteilt werden [TBMM04]:

1. Primäre Komponenten
    - Einfache Typdefinitionen
    - Komplexe Typdefinitionen
    - Attributdeklarationen
    - Elementdeklarationen
2. Sekundäre Komponenten
    - Attribut-Gruppen-Definitionen
    - Definitionen von Identitätsbeschränkungen
    - Model-Gruppen-Definitionen
    - Notationsdefinitionen
3. "Helfer" Komponenten
    - Annotationen
    - Model-Gruppen
    - Partikel
    - Wildcarts
    - Attribut Verwendung

Dadurch erweist sich XML Schema als gut geeignet zur Definition einer XML basierten Beschreibungssprache.

4.3.2 eXtensible Game Description Language (xgdl)

Unter Verwendung von XML Schema und basierend auf den theoretischen Untersuchungen der virtuellen Parameter wurde eine auf XML basierende Beschreibungssprache für virtuellen Kontext entwickelt, die e**X**tensible **G**ame **D**escription **L**anguage (**xgdl**). Die xgdl ist dreistufig aufgebaut: die vier **Kontextkategorien** (Generelle Informationen *General*, Lokationsinformationen *Location*, Charakterinformationen *Character* und Statusinformationen *Status*) enthalten **Parametersets** in denen die **virtuellen Parameter** zusammengefasst werden.

Abbildung 4.11 zeigt den Basis-Parametersatz für MOGs (siehe auch Anhang A.2). Dieser Basissatz generalisiert die virtuellen Parameter mindestens für alle virtuellen Welten vom Typ MOG (kann aber auch andere virtuelle Welten abdecken) und ist gleichzeitig sowohl genre- als auch spielspezifisch erweiterbar (siehe Beispiele in Abbildung 4.12 mit Erweiterungen für FPS und MMOGs). Erweiterte Parametersätze enthalten jeweils den Basis-Parametersatz. Durch die Erweiterung von allgemeinen Parametersätzen hin zu spezialisierten wird die Interoperabilität sichergestellt und eine gemeinsame Basis geschaffen. Die Erweiterbarkeit ermöglicht, dass zusätzlich zu der Basisbeschreibung noch genre- oder spielspezifische Charakteristika abgebildet werden können. Zum Beispiel enthält die Lokationsinformation des FPS Parametersatzes zusätzliche Informationen über die Map (eine feste Gebietsinstanz), auf der sich ein Charakter gerade befindet (*Map information*), und der MMOG Parametersatz enthält zusätzlich Informationen über das Gebiet (Teil der Spielwelt), in dem sich der Charakter gerade aufhält (*Area Information*). Durch die Verwendung von XML Schema ist es außerdem möglich, verschiedene Parametersätze wieder zu kombinieren, um dann z.B. einen MMOFPS Parametersatz ableiten zu können. Der abgeleitete Parametersatz beinhaltet dann die Parameter des MMOG-Parametersatzes und die des

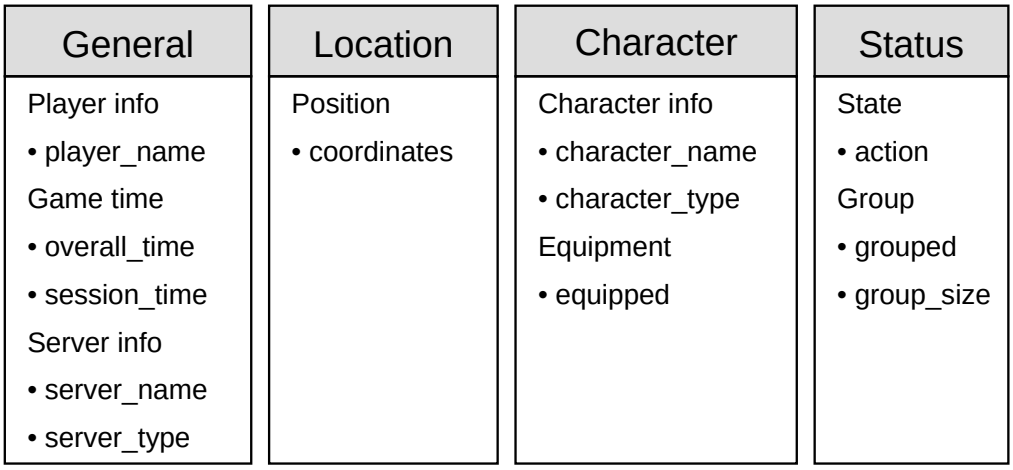

Abbildung 4.11: Generische Parameterklassen und Parameter für MOGs

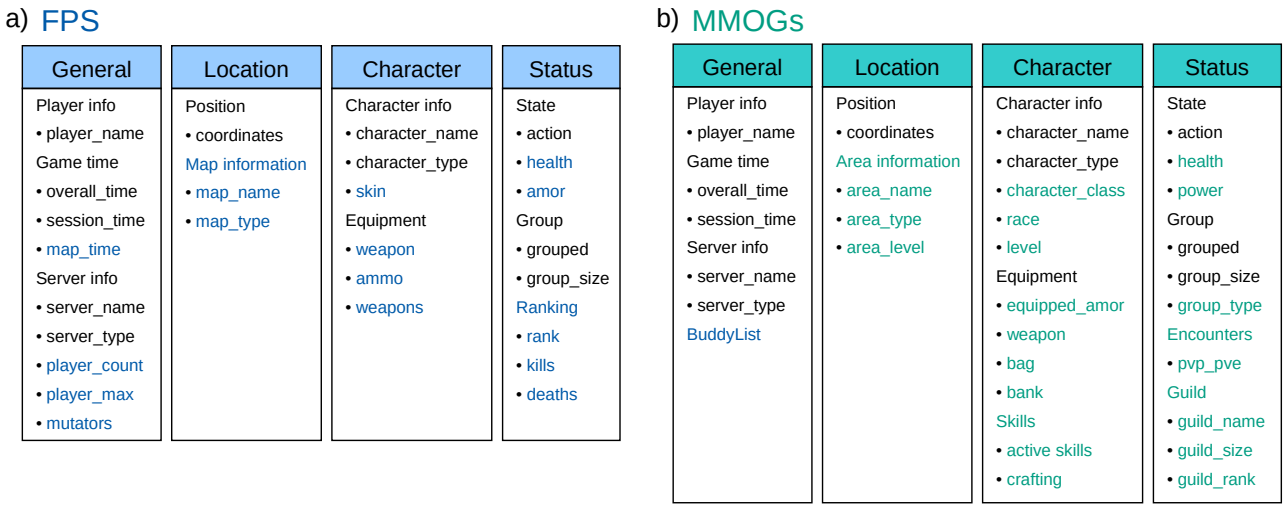

Abbildung 4.12: Beispiele für Parameterklassen und Parameter erweitert für verschiedene MOG Genres: a) First Person Shooter (FPS) b) Massively Multiplayer Online Games (MMOGs)

FPS-Parametersatzes, wobei die Parameter, die in beiden Parametersätzen vorkommen, nur einmal in den abgeleiteten MMOFPS-Parametersatz aufgenommen werden.

Virtuelle Parameter, wie die Positionskoordinaten eines virtuellen Charakters in der virtuellen Welt, sein Equipment oder die Aktion, die er gerade in der virtuellen Welt ausführt, werden durch die xgdl abgebildet und dadurch in ein austauschbares Datenformat überführt (siehe Beispiel in Abbildung 4.13).

Xgdl ist flexibel und erweiterbar, da sie entsprechend der Definition für Parametersätze erweitert werden kann, um zusätzliche virtuelle Parameter aufzunehmen. Durch das modulare Konzept ist es darüber hinaus möglich, neue Parametersätze einzubinden (damit können zukünftig entwickelte MOGs und weitere virtuelle Welten integriert werden). Durch die Verwendung von xgdl wird eine adäquate Beschreibung von virtuellem Kontext ermöglicht, da die Definition von xgdl auf der Analyse der in virtuellen Welten verwendeten virtuellen Parametern basiert. Dies ist die Basis dafür, in-game Kontextinformationen in einem allgemeinen Format austauschbar zu machen und abstrakte Interfaces für die Verwendung in verschiedenen virtuellen Welten zu definieren.

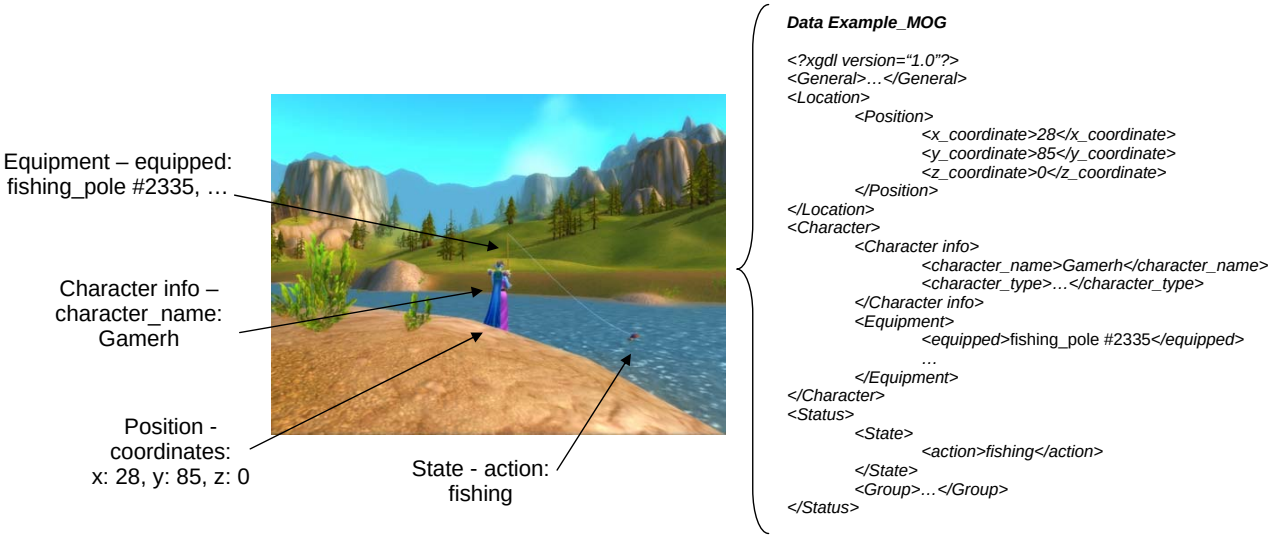

Abbildung 4.13: MOG Beispiel für Parameter in virtuellen Umgebungen und deren Beschreibung

4.4 Verwendung von virtuellem Kontext

Virtuelle Welten (vergleiche Kapitel 3.2.1) sind komplexe Umgebungen in denen die Benutzer mit der virtuellen Welt interagieren, in denen aber auch die Interaktion mit den anderen Benutzern der virtuellen Welt stattfindet. Um den Benutzer in seiner Interaktion mit der virtuellen Welt zu unterstützen, bietet eine virtuelle Welt verschiedene einfache Dienste an. Ein solcher Dienst ist beispielsweise eine kleine Ausschnittskarte (*Minimap*), die dem Benutzer seine unmittelbare Umgebung und seine Ausrichtung anzeigt. Zusätzlich können weitere Dienste diese Karte verwenden, um darauf bestimmte Personen zu markieren (zum Beispiel Händler) oder um die Richtung anzuzeigen, in der das Ziel der aktuellen Aufgabe (*Quest*) des Charakters zu finden ist. Auch die Interaktion der Benutzer untereinander wird in der Regel durch einfache Dienste von der virtuellen Welt unterstützt. Beispielsweise sind in-game Gruppen eine Form der Unterstützung. Verschiedene Charaktere, die sich in einer in-game Gruppe zusammenschließen, bekommen zusätzliche Gruppen-Dienste zur Verfügung gestellt, wie die Anzeige von Zusatzinformationen über die anderen Gruppenmitgliedern oder einen zusätzlichen Gruppen-Chat.

Die Möglichkeit, zusätzliche Dienste anzubieten, wird immer wichtiger, da die Anforderungen aktueller MOGs an den Spieler immer komplexer werden. Die grundlegenden Dienste (wie *Minimap* oder Gruppen) sind bereits zu einem integralen Bestandteil der MMOGs geworden. Aufgrund der steigenden Notwendigkeit dieser Zusatzdienste und wegen der immer populärer werdenden Möglichkeiten, sich als Spieler eigene Dienste erstellen zu können, steigt deren Anzahl an und eine Auswahl von Diensten wird notwendig. Um die immer komplexeren Aufgaben in MOGs zu bewältigen, entstehen auch immer spezialisiertere Dienste. Die Benutzerunterstützung durch spezialisierte Dienste muss jedoch flexibel sein, da im Gegensatz zu allgemeingültigen Diensten (wie die *Minimap*), spezialisierte Dienste nur in bestimmten Situationen für den Spieler überhaupt relevant sind und ansonsten ein Übermaß an Diensten entsteht, die den Spieler dann im schlimmsten Falle mehr behindern als unterstützen.

Die durchgeführte Analyse hat gezeigt, dass in allen MOGs die Kooperation zwischen Spielern notwendig für ein erfolgreiches Gruppenspiel ist. Durch die Kooperation entstehen aber auch verschiedene Anforderungen. So müssen Aufgaben und Rollen (Meta- und Charakterrollen) unter den Spielern verteilt werden und ein koordiniertes Vorgehen muss geplant und durchgeführt werden. Damit jedoch eine solche Kooperation funktioniert, müssen unter anderem geeignete Kommunikationsformen eingesetzt werden. Für Dienste, die die Gruppenkooperation in MOGs unterstützen sollen, stellen vor allem die entstehenden komplexen Szenarien und die hohe Dynamik Herausforderungen dar. Dabei ist zum Beispiel vor

einer Begegnung die Unterstützung der Rollenaufteilung wichtig. Während einer Begegnung wird diese nicht mehr benötigt, dafür jedoch eine Unterstützung von schnellen Absprachen, beispielsweise durch einen Sprach-Kommunikationsdienst. Ein Sprach-Kommunikationsdienst, der das Gruppenspiel passend zur Situation unterstützen kann, muss sich dynamisch an die wechselnden Anforderungen anpassen. So ist es sinnvoll, dass zu Beginn alle Gruppenmitglieder zusammen in einem gemeinsamen Kommunikationskanal sind, um gemeinsame Absprachen treffen zu können und die Rollen aufzuteilen. Während einer Begegnung ist es jedoch wünschenswert, dass sich Teilgruppen einzeln absprechen können, um Teilaktionen zu koordinieren. Um solche Absprachen zu ermöglichen, ohne die anderen Teilgruppen zu stören, sind unterschiedliche Kommunikationskanäle notwendig, wobei die Aufteilung der Gruppenmitglieder auf die Teilgruppen je nach Situation unterschiedlich sein kann und somit eine dynamische Zuweisung der Spieler zu den verschiedenen Kommunikationskanälen erfolgen muss (siehe 6.3.1).

Über die Gruppenspiel-Situation hinaus gibt es weitere Situationen in MOGs, die durch Dienste unterstützt werden können. Solche Situation umfassen die Gruppenfindung, den Handel von Gütern oder Dienstleistungen, den Abbau von Rohstoffen aus der virtuellen Umgebung oder die Ausübung von Berufen. Dienste, die eine Unterstützung des Benutzers anbieten, können Informationsdienste, Kommunikationsdienste oder Kooperationsdienste sein.

Informationsdienste, stellen dem Benutzer zusätzliche Informationen zur Verfügung oder stellen vorhandene Informationen anders dar. Dabei kann es sich um allgemeine Informationsdienste wie Guidelines oder Howto's oder um spezialisierte Informationsdienste, wie den "Fisherman's Friend" (siehe auch 6.3.3), handeln.

Kommunikationsdienste erleichtern die Absprache oder die Kommunikation verschiedener Benutzern. Zum Beispiel kann einem Benutzer ein Dienst zur Unterstützung der Sprach-Kommunikation angeboten werden, wenn er sich in der virtuellen Welt einer Gruppe anschließt. Verwendet der Benutzer den Dienst, dann kann dieser ihn automatisch in den Kommunikations-Kanal der Gruppe verbinden. Ohne virtuell kontextbasierte Dienste muss ein Spieler dafür zuerst die login-Daten von seinen Mitspieler via in-game Text-Chat erfragen, danach aus dem Spiel zum externen Voice-Chat Tool wechseln, sich mit dem passenden Server verbinden, manuell in den passenden Kanal wechseln und nachdem die Sprachkommunikation eingerichtet ist, wieder zum eigentlich Spiel zurückkehren.

Kooperationsdienste unterstützen die Zusammenarbeit von Spielern. Ein konkretes Beispiel ist ein Dienst, der die gemeinsame Dokumentation der virtuellen Umgebung unterstützt. Ein Dokumentationsdienst kann die Informationen verwerten, die ein Spieler während seines Aufenthalts in der Spielwelt bereitstellt. Mit dem "*lokationsbasierten Annotationsdienst*" (siehe auch 6.3.4), ist es etwa möglich, dass der Spieler Annotationen in der virtuellen Umgebung hinterläßt. Diese Annotationen können dann mit anderen Spielern geteilt oder außerhalb des Spiels veröffentlicht werden (z.B. in einem Wiki).

Im Folgenden wird das Konzept für eine situationsabhängige Unterstützung der Interaktion in virtuellen Welten vorgestellt, welches die durch die Interaktion in virtuellen Welten entstehenden Herausforderungen umsetzt. Im Anschluß folgt die Abgrenzung des entwickelten Konzepts der virtuell kontextbasierten Dienste von dem Konzept der kontext-bewussten Dienste aus dem verwandten Bereich der kontext-bewussten Systeme (siehe 3.1).

4.4.1 Virtuell kontextbasierte Dienste

Eine statische Nutzung aller verfügbaren Dienste ist nicht nur sehr unübersichtlich, sondern stellt auch ein Überangebot für den Benutzer dar, da immer nur bestimme Dienste für den Benutzer in bestimmten Situationen interessant sind. Befindet sich ein Benutzer beispielsweise in einer Konfrontation mit virtuellen Gegnern, so benötigt er keinen Dienst, der ihn beim Kauf und Verkauf von Gegenständen unterstützt. Um den Herausforderungen gerecht zu werden, muss das Dienstangebot eingeschränkt werden, sodass der Benutzer nur eine Auswahl an Dienste angeboten bekommt, die der aktuellen Situation des Benutzers entsprechen, und somit für ihn nützlich sind. Ebenso muss eine Anpassung der Dienste

entsprechend der Dynamik sich ändernder Spielsituationen vorgenommen werden. Das Ziel dieser Arbeit ist es, situationsabhängig unterstützende Dienste, die sich an die virtuelle Situation eines Spielers anpassen, anbieten zu können und somit den Spieler in seiner aktuellen Situation in der virtuellen Umgebung adäquat unterstützen zu können.

Wie die Analyse der Interaktion in virtuellen Welten zeigt, hat ein virtueller Charakter immer eine bestimmte Menge an Interaktionen, die er ausführen kann (vergleiche 4.1.2). Diese sind abhängig von der Klasse des Charakters und der Charakterentwicklung. Die Vereinigungsmenge aller Interaktionen, die ein virtueller Charakter innerhalb einer virtuellen Welt ausführen kann, korreliert mit allen möglichen Diensten, die einem Benutzer der virtuellen Welt angeboten werden können. Jedoch können nicht alle Interaktionen in jeder Situation ausgeführt werden. Es existieren zwei Klassen von Parametern (reale und virtuelle Parameter), die die möglichen Interaktionen einschränken und die Entscheidungen des Benutzers beeinflussen. Die durchgeführten Untersuchungen haben gezeigt, dass die virtuelle Situation der virtuellen Repräsentation eines Benutzers, also der virtuelle Kontext eines Benutzers, über die virtuellen Parameter definiert ist (vergleiche Abschnitt 4.2.2). Das abgeleitete generische Modell der Interaktion in virtuellen Welten (vergleiche Abschnitt 4.1.3), das die Untersuchungen der Interaktionsmöglichkeiten zusammenfasst, und die Kontextbeschreibung basierend auf virtuellen Parametern (vergleiche Abschnitt 4.3.2), bilden die Grundlage für eine Unterstützung der Interaktion in virtuellen Welten durch situationsabhängige Dienste.

Indem man virtuellen Kontext beschreibbar und nutzbar macht, wird es möglich, Dienste mit diesem Kontext zu verknüpfen. Analog zu den kontext-bewussten Diensten, die sich auf realen Kontext beziehen, können dadurch Dienste angeboten werden, die auf virtuellem Kontext basieren. In Abgrenzung zu realen kontext-bewussten Diensten, die Dienste abhängig vom realen Benutzerkontext ausführen oder anpassen (*benutzerabhängige Dienstausführung und Dienstanpassung* - vergleiche 3.1.3), werden Dienste, die sich auf virtuellen Kontext beziehen, hier definiert als *virtuell kontextbasierte Dienste* (***V**irtual **C**ontext **B**ased **S**ervices* - **VCBS**).

Definition 4.2 *Ein **virtuell kontextbasierter Dienst** ist ein Dienst, der auf Basis von virtuellem Kontext, situationsabhängig angeboten und / oder angepasst werden kann.*

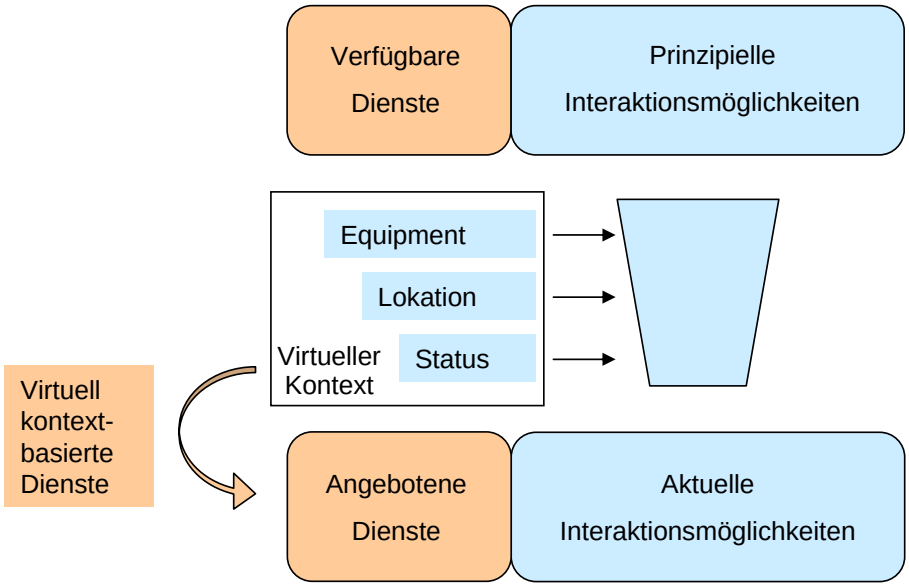

Abbildung 4.14: Virtuell kontextbasierte Dienste

Abbildung 4.14 zeigt die Verknüpfung von Diensten und Interaktionen. Alle Dienste, die prinzipiell verfügbar sind (*verfügbare Dienste*) sind analog zu der Menge aller Interaktionsmöglichkeiten (*prinzipielle*

Interaktionsmöglichkeiten). Basierend auf dem virtuellen Kontext werden dann aus allen verfügbaren Diensten, diejenigen bestimmt, welche der virtuellen Situation entsprechen. Analog zu den *aktuellen Interaktionsmöglichkeiten* ergibt sich somit die Teilmenge der *angebotenen Dienste*. Die Entscheidung welche Dienste dem Benutzer offeriert werden ist abhängig von seiner aktuellen virtuellen Situation, nur solche Dienste, die seiner virtuellen Situation entsprechen, werden dem Benutzer auch angeboten.

4.4.2 Virtuell kontextbasierte Dienste und (real) kontext-bewusste Dienste

Bei kontext-bewussten Diensten werden Kontextinformationen verwendet, um Dienste bereitzustellen oder anzupassen (siehe auch Kapitel 3.1). Dabei werden die Kontextinformationen einer Person für die benutzerbezogene Dienstanpassung verwendet. Die Gewinnung der Kontextinformationen erfolgt über Sensordaten, die im Umfeld der Person erfasst werden, und aus denen dann die benötigten Kontextinformationen abgeleitet werden. Für die Erfassung von Sensordaten sind geeignete Sensoren notwendig, die installiert werden müssen, um überhaupt Informationen über die aktuelle Situation einer Person erhalten zu können. Für das Angebot von kontext-bewussten Diensten bedarf es also einer entsprechenden Umgebung, die mit Sensoren überwacht werden kann. Ein Beispiel ist ein kontext-bewusster Dienst, der in einem Museum eingesetzt wird, um den Besuchern abhängig von ihrer Position zusätzliche Informationen anzubieten. Dazu bedarf es beispielsweise eines Gerätes, das der Benutzer mit sich führt und welches seine Position und eventuell auch seine Blickrichtung an den kontext-bewussten Museums-Informationsdienst weitergibt. Auf Basis der Positionsangaben kann der Dienst nun die Stelle im Museum bestimmen, an der sich der Besucher gerade befindet, und dem Besucher zusätzliche Informationen zu dem Exponat, das er gerade betrachtet, anbieten. Ein weiteres Beispiel sind die sogenannten smarten Räume. Dabei handelt es sich um besonders ausgestattete Räumlichkeiten, in denen kontext-bewusste Dienste zur Verfügung gestellt werden können. Auch in diesem Fall haben die Personen, die diese Dienste nutzen möchten, in der Regel ein spezielles Gerät bei sich, das Informationen (wie die Position des Person) an Applikationen weitergibt, die Dienste zur Verfügung stellen. Zusätzlich können über Sensoren, die im smarten Raum angebracht sind, wie beispielsweise Kameras oder Bewegungsmelder, weitere Sensordaten erfasst werden. Aus allen erfassten Sensordaten werden dann wiederum Kontextinformationen abgeleitet, zum Beispiel "Person A betritt gerade den smarten Raum" oder "Person A und Person B und drei weitere Personen sitzen zusammen am Tisch". Kontext-bewusste Dienste können nun diese Informationen nutzen. Im ersten Beispiel kann ein kontext-bewusster Beleuchtungsdienst das Licht einschalten, sobald eine Person den Raum betritt und im zweiten Beispiel kann für Person A abgeleitet werden, dass sie sich in einem Meeting befindet und ein kontext-bewusster Telefonie-Dienst kann ankommende Anrufe umleiten, damit die Person nicht gestört wird.

Kontextinformationen einer Person, die an einem Computer arbeitet, können über Software-Sensoren erfasst werden. Diese Software-Sensoren können beispielsweise das Programm, das die Person gerade verwendet, ermitteln oder ihrer Aktivität anhand von Tastatur- und Mauseingaben ableiten.

Kontext-bewusste Dienste sind immer auf die Erfassung von Sensordaten angewiesen und funktionieren nur wenn die notwendigen Sensoren zur Verfügung stehen. Es wird daher nie möglich sein, alle Kontextinformationen einer Person zu erfassen, da dafür eine unüberschaubar große Anzahl an Sensoren benötigt werden würde. Darüber hinaus ist es erforderlich, dass aus den erfassten Sensordaten auch die für einen Dienst notwendigen Kontextinformationen abgeleitet werden können. Sensordaten können je nach Sensor in ihrer Verfügbarkeit und Genauigkeit sehr unterschiedlich sein und müssen interpretiert werden, um den Kontext der Person daraus ableiten zu können. Ein Beispiel dafür ist die Abbildung von GPS-Koordinaten auf Räume in einem Gebäude. Kontext-bewusste Dienste sind somit Dienste, die sich spezieller Kontextinformationen einer Person bewusst sind und auf Basis dieser Informationen angeboten und gesteuert werden können.

Das Grundkonzept der virtuell kontextbasierten Dienste ist das Angebot von Diensten sowie deren Anpassung bezüglich der virtuellen Situation des Benutzers. Dadurch ist es möglich, den Benutzer passend

zu seiner aktuellen Situation zu unterstützen. Die "Sensordaten" der kontextbasierten Dienste sind die virtuellen Parameter der virtuellen Welt. Diese stehen immer und vor allem auch immer in der gleichen Qualität zur Verfügung. Darüber hinaus kann über die Summe aller virtuellen Parameter die virtuelle Situation des Charakters vollständig beschrieben werden, wodurch der virtuelle Kontext des Benutzers genau ermittelt werden kann. Auf Basis der verfügbaren Parameter stellt auch der Client einer virtuellen Welt die virtuelle Situation des Benutzers dar. Kontextbasierte Dienste können somit in jeder Situation, in der sich der virtuellen Charakter des Benutzers in der virtuellen Welt befindet, angeboten werden und sind nicht auf bestimmte Gebiete begrenzt, in denen geeignete Sensoren zur Verfügung stehen. Sie können die virtuelle Situation, in der sie angeboten werden, oder die Parameter, nach denen sie angepasst werden sollen, genau spezifizieren, da sie prinzipiell auf alle notwendigen Kontextinformationen Zugriff haben und nicht durch eine begrenzte Auswahl an Sensordaten beschränkt sind.

Zusammenfassend sind beide, sowohl kontext-bewusste, als auch kontextbasierte Dienste, solche Dienste, die auf Grundlage von Kontextinformationen angeboten und gesteuert werden können. Der entscheidende Unterschied zwischen beiden ist jedoch die Art der verwendeten Kontextinformationen. Die "realen" Kontextinformationen, die bei kontext-bewussten Diensten verwendet werden, müssen aus Sensordaten abgeleitet werden, sind somit von Verfügbarkeit und Qualität der Sensordaten abhängig und können den Kontext einer Person nicht vollständig abbilden. Die virtuellen Kontextinformationen, die bei kontextbasierten Diensten verwendet werden, sind die virtuellen Parameter des virtuellen Charakters. Diese sind immer und an jeder Stelle der virtuellen Welt in gleicher Qualität verfügbar und können die Situation des virtuellen Charakters und somit den virtuellen Kontext des Benutzers vollständig beschreiben.

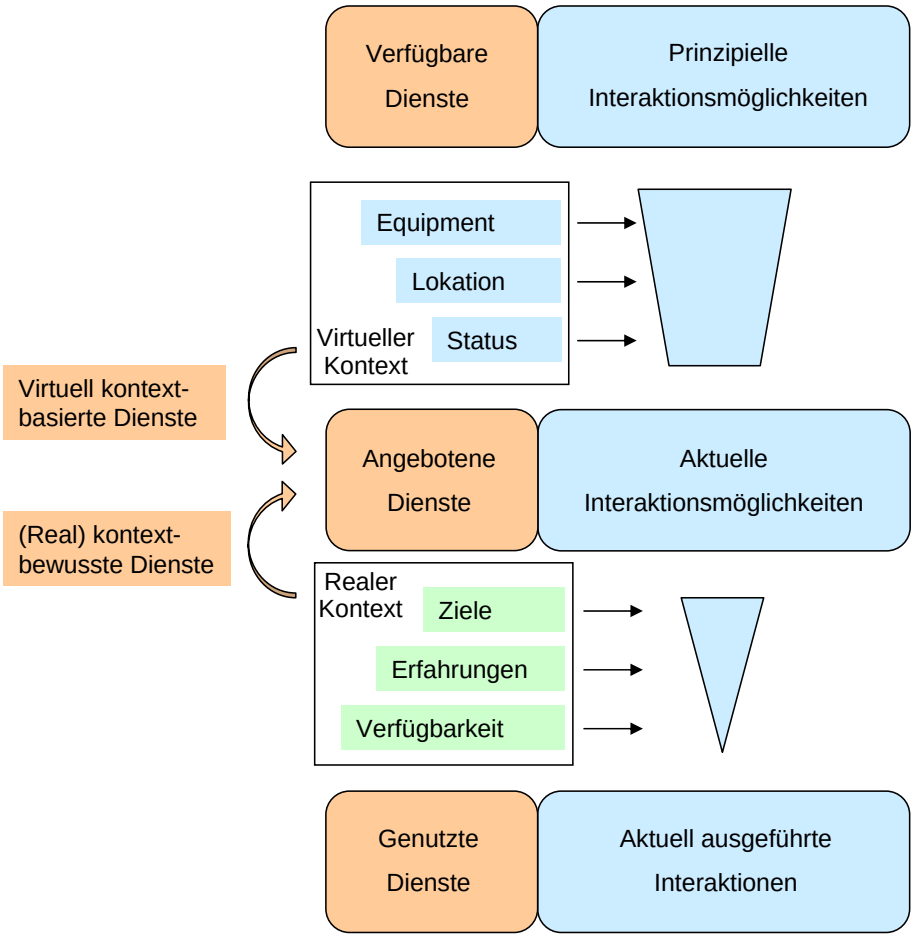

Abbildung 4.15: Kontextbasierte und kontext-bewusste Dienste

Es ist möglich, beide Arten von Diensten zu kombinieren (siehe Abbildung 4.15). Durch eine Kombination von virtuellen und realen Kontextinformationen sowie die Verwendung von kontext-bewussten und virtuell kontextbasierte Diensten, kann zusätzlich zum virtuellen der reale Kontext des Benutzers für die Bestimmung der *angebotenen Dienste* verwendet werden. Dadurch ergeben sich zusätzliche Möglichkeiten. Beispielsweise kann das Serviceangebot passend zur virtuellen Situation und zusätzlich angepasst auf den Benutzer zugeschnitten werden. Dabei kann der reale Kontext eines Benutzers sowie das Nutzerverhalten für eine weitere Anpassung der angebotenen Dienste an die Bedürfnisse und Vorlieben des Benutzers verwendet werden. Darüber hinaus wird die Beziehung der *aktuell ausgeführten Interaktionen* zu den verwendeten Diensten (*genutzte Dienste*) dargestellt. Wie auch die Auswahl der Interaktion abhängig vom Benutzer, seinen Eigenschaften und seinem realen Kontext erfolgt, so ist letztendlich auch die Verwendung der angebotenen Dienste davon abhängig.

Dabei sind, wie auch bei den kontext-bewussten Diensten, verschiedene Stufen der Dienstautomatisierung möglich (vergleiche Kapitel 3.1.3). Sowohl die Ausführung der angebotenen Dienste, als auch die Anpassung der von den Diensten bereitgestellten Informationen und Funktionen, kann automatisch oder manuell erfolgen.

Diese Arbeit schafft die Grundlagen für eine Verwendung von kontextbasierten Diensten in virtuellen Welten. Auf diesen Grundlagen und verwandten Ansätzen basierende Erweiterungen und Kombinationen, wie die Kopplung von virtuellem und realem Kontext ergeben weiterführende Forschungsfragen, die über die hier adressierten Grundlagen hinausgehen.

4.5 Zusammenfassung

Virtuell kontextbasierte Dienste (VCBS) können basierend auf dem virtuellen Kontext eines Benutzers angeboten und angepasst werden. Sie verwenden Informationen über den virtuellen Kontext des Benutzers und bieten ihm Dienste an, die in seiner virtuellen Situation nützlich sind. Diese Dienste sollen den Benutzer innerhalb der virtuellen Umgebung unterstützen. Für die Verwendung von virtuellem Kontext für Dienste finden sich in der Literatur keine relevanten verwandten Arbeiten. Dies ist jedoch essentiell für eine Benutzerunterstützung in virtuellen Umgebungen und bildet die Basis für eine Integration von Diensten in virtuelle Welten.

Virtuell kontextbasierte Dienste können in allen Formen virtueller Welten, wie sie heutzutage existieren, und in ganz unterschiedlichen Szenarien eingesetzt werden. Im weiteren Verlauf dieser Arbeit wird ihr Einsatz im Rahmen von Multiplayer Online Games behandelt. Wie eingangs (vergleiche Kapitel 3.2.2) beschrieben, handelt es sich bei den MOGs um die heute am weitesten verbreitete, interessanteste und bedeutendste Form virtueller Welten. Ziel des Einsatzes von virtuell kontextbasierten Diensten in MOGs ist es, neben der dynamischen Benutzerunterstützung, externe kontextbasierte Dienste in die virtuellen Umgebungen zu integrieren und damit eine Verknüpfung von virtuellen Welten und anderen Internet-Applikationen zu schaffen und einen Informationsaustausch zu ermöglichen. Durch virtuell kontextbasierte Dienste werden Möglichkeiten für eine Unterstützung der Benutzer während ihres Aufenthalts in einer virtuellen Welt und für eine aktive Beteiligung der Gaming Communities bei der Benutzerunterstützung geschaffen. Eine detaillierte Modellierung der betrachteten Problemstellung, das Design und die Entwicklung der technischen Umsetzung von virtuell kontextbasierten Diensten sowie deren Realisierung und Evaluation finden sich in den folgenden Kapiteln.

5 Informationsaustausch zwischen Spielen und Internet-Applikationen durch VCBS

Zur Realisierung der Unterstützung von Spielern innerhalb virtueller Welten durch externe Dienste wird eine Verknüpfung von Spiel und Spielumfeld benötigt. Die Verknüpfung von Diensten, die von externen Internet-Applikationen bereitgestellt werden mit einem Spiel erfordert die Realisierung eines kontrollierten Informationsaustauschs und die Integration der externen Dienste in die virtuelle Welt.

Wie zu Begin der Arbeit vorgestellt (vergleiche Kapitel 2), existiert neben den Spielen eine große Anzahl von Internet-Applikationen, die die Spieler zusätzlich zum Spiel nutzen. Zwischen einem Spiel und diesen Applikationen im Spielumfeld ist zumeist kein Austausch von Informationen möglich. Um Cheating oder die Verwendung von unerlaubten Hilfen oder unfairen Vorteilen zu unterbinden, erlauben viele Spiele nicht, Informationen mit der "Außenwelt" auszutauschen oder spielinterne Informationen (in-game Informationen) außerhalb des Spiels bereitzustellen.

Die Spielwelten von MOGs sind deshalb oftmals isolierte Applikationen. Daher benötigt ein Benutzer, wie in Kapitel 3.3.3 dargestellt, verschiedene Programme (beispielsweise *Browser* oder *Voice-Tools*), um spielbezogene Anwendungen (wie ergänzende *Voice-Chats*) nutzen zu können oder um auf spielbezogenen Informationen (beispielsweise auf einem *Community-Portal* oder in einem *Wiki*) zugreifen zu können (siehe Abbildung 5.1). Nur in einzelnen Fällen gibt es einen Informationsaustausch zwischen Spielen und anderen Internet-Applikationen, der in der Regel jedoch nur eine Richtung des Informationsflusses unterstützt und sehr beschränkt ist.

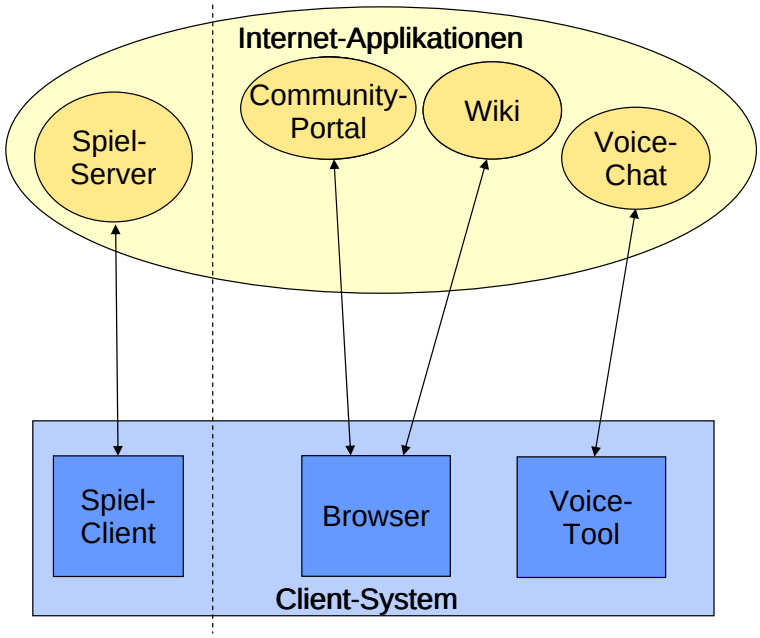

Abbildung 5.1: Verschiedene Applikationen die zusätzlich zum Spielen verwendet werden

Ein kontrollierter Informationsaustausch zwischen Spiel und Spielumfeld kann auf Basis des Konzepts der virtuell kontextbasierten Dienste umgesetzt werden. Externe Dienste können somit, basierend auf der aktuellen Situation des Spielers, direkt in das Spiel integriert werden.

Im Folgenden werden zunächst die Anforderungen an einen Informationsaustausch zwischen Spielen und ihrem Umfeld dargestellt, bestehende Ansätze werden vorgestellt und bewertet. Im Anschluss werden die gewählte Lösungsstategie und das Design des gewählten Ansatzes detailliert vorgestellt. Dies

beinhaltet die konzeptuelle Darstellung des entwickelten VCBS-basierten Middleware-Ansatzes für die Realisierung des Informationsaustauschs und seiner Eigenschaften, sowie die Darstellung der Integration von Diensten in virtuelle Welten und weiterer Aspekte aus den Bereichen Sicherheit und Benutzerfreundlichkeit. Die Umsetzung der entsprechenden Middleware-Lösung wird dann im nachfolgenden Kapitel beschrieben.

5.1 Anforderungen an den Informationsaustausch

Der grundlegende zu realisierende Informationsaustausch wird in Abbildung 5.2 dargestellt und umfasst die folgenden Funktionalitäten. Spielintern sind Informationen über das Spielgeschehen zu erfassen. Nach der Erfassung der Informationen im Spiel (*in-game Informationen*) müssen diese über eine geeignete Schnittstelle (*Spiel-Schnittstelle*) an das Spielumfeld übertragen werden. Im Spielumfeld können diese in-game Informationen dann bereitgestellt und genutzt werden. *Externe Informationen*, die von der *Community* erstellt oder aus den in-game Informationen generiert wurden, müssen wiederum über die *Spiel-Schnittstelle* an das Spiel übertragen und dem Benutzer innerhalb des Spieles bereitgestellt werden.

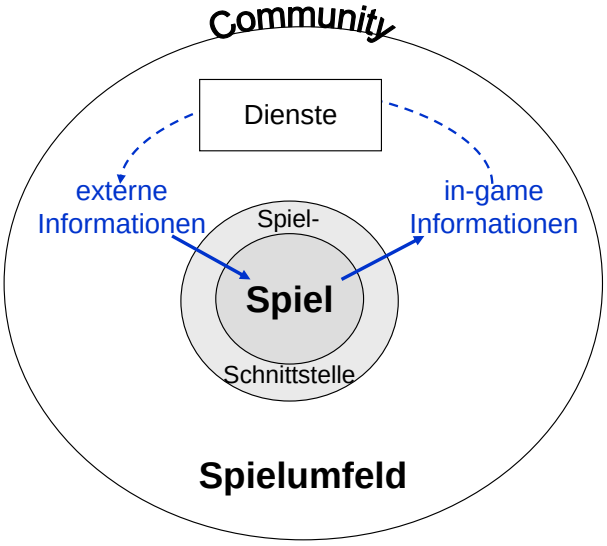

Abbildung 5.2: Informationsfluss zwischen Spiel und Spielumfeld

Bei einer näheren Betrachtung des Szenarios handelt es sich bei den externen Applikationen um *Dienste*, die zum einen Informationen aus einer virtuellen Welt außerhalb der virtuellen Welt darstellen und zum anderen dem Benutzer Funktionen und Informationen bereitstellen, während er sich innerhalb einer virtuellen Welt aufhält. Die Nutzung solcher zusätzlichen Dienste innerhalb der virtuellen Welt unterliegt speziellen Beschränkungen. Vor allem darf das eigentliche Spielen durch die Dienste nicht beeinträchtigt werden. Vielmehr sollten die Dienste den Benutzer beim Spielen unterstützen. Dem Spieler sind, wie in Kapitel 4.4.1 motiviert, nicht jederzeit alle Dienste zur Verfügung zu stellen. Die Bereitstellung von Diensten soll situationsbedingt, in Abhängigkeit vom virtuellen Kontext des Benutzers, erfolgen. Der virtuelle Kontext wird durch in-game Informationen erfasst. Basierend auf den erhaltenen in-game Informationen erfolgen dann Angebot und Bereitstellung der Dienste. Dienstangebot und Bereitstellung müssen innerhalb des Spiels geeignet dargestellt und eine entsprechende Dienstnutzung ermöglicht werden. Darüber hinaus soll es der Community möglich sein, in-game Informationen auch außerhalb des Spiels nutzen zu können, sowie eigene Dienste zur Verfügung zu stellen. Verschiedene Untersuchungen haben gezeigt, dass mehr Informationen, als momentan verfügbar sind, für die Spieler zugreifbar und für die Gaming Community offen sein müssen (siehe 3.3.3). Auf der anderen Seite gibt es natürlich auch Gründe, warum die Spieleentwickler den Zugang zu ihren Spielen einschränken. In erster Linie wollen

Sie somit ihre Spiele schützen (Cheating) und sicherstellen, dass die Spieler ein ungetrübtes Spielerlebnis haben.

5.1.1 Funktionale Anforderungen

Ein kontrollierter Informationsaustausch, der beide Richtungen des Informationsflusses unterstützt, kann durch die Umsetzung virtuell kontextbasierter Dienste und die Nutzung der virtuellen Kontextinformationen ermöglicht werden. Die grundlegenden Anforderungen bestehen in der Realisierung eines *Informationsaustauschs* in beide Richtungen sowie der kontextbasierten Bereitstellung von Diensten (*virtuell kontextbasierte Dienste*), unabhängig von einer bestimmten virtuellen Welt und der angebundenen Internet-Applikationen (*generisches Konzept*).

Informationsaustausch

Die beiden Informationsflüsse, die sich aus der vorgestellten Modellierung ergeben, sind zum einen die Erfassung von in-game Informationen und deren Übertragung vom Spiel in dessen Umfeld und zum anderen die Bereitstellung von Diensten aus dem Spielumfeld, innerhalb des Spiels, durch die Übertragung *externer Informationen* (Servicedaten). Für die Realisierung des Informationsaustauschs wird eine generische Schnittstelle für den Informationsaustausch mit der virtuelle Welt (*Spiel-Schnittstelle*) benötigt, die es ermöglicht, in-game Informationen im Spielumfeld abzurufen sowie externe Informationen in der virtuellen Welt zugreifbar zu machen.

Virtuell kontextbasierte Dienste

Für die Realisierung der Anbindung externer Dienste auf Basis des Konzepts der virtuelle kontextbasierten Dienste, müssen die virtuellen Kontextinformation erfasst und verarbeitet und die externen Dienste in die virtuelle Welt integriert werden. Die *Erfassung der Kontextinformationen* eines virtuellen Charakters (virtueller Kontext) erfolgt durch die in-game Informationen. Um situationsabhängig Dienste basierend auf virtuellem Kontext anbieten zu können, müssen die erfassten Kontextinformationen weiterverarbeitet werden, wobei die *Kontextverarbeitung* die Auswertung der Kontextinformationen und deren Verwendung zur Bestimmung passender Dienste beinhaltet. Die besondere Herausforderung hierbei liegt im Matching von Diensten und virtuellen Situationen. Hierfür wird eine geeignete Beschreibung der Dienste benötigt. Für Angebot und Nutzung von externen Diensten innerhalb von virtuellen Welten müssen die Dienste in die virtuelle Umgebung integriert werden. Die *Integration der Dienste* in die virtuelle Welt ermöglicht das Angebot der Dienste, die Dienstinteraktion und die Dienstnutzung inklusive der vom Dienst bereitgestellten Informationen (Servicedaten) innerhalb der virtuellen Welt.

Generisches Konzept

Das Konzept soll generisch eingesetzt werden, also weder spiel- noch applikationsspezifische Elemente enthalten. Die erfassten Kontextinformationen müssen zum einen geeignet an die von verschiedenen Internet-Applikationen bereitgestellten externen Dienste verteilt werden. Zum anderen müssen die von den Diensten erzeugten Servicedaten gesammelt werden, sodass eine Übertragung der Servicedaten unabhängig von den verwendeten Diensten erfolgen kann. Für die *Verteilung der Kontextinformationen* an die externe Dienste und die *Sammlung der Servicedaten*, sowie deren Weiterleitung und Integration ins Spiel, muss neben einer generischen Spiel-Schnittstelle eine geeignet *Dienst-Schnittstelle* zur Verfügung gestellt werden.

Sowohl für die Realisierung des Informationsaustauschs, wie auch für die Bereitstellung externer virtuell kontextbasierter Dienste, ist eine **Integration von Diensten** in die virtuelle Welt nötig. Diese Integration beinhaltet die integrierte Darstellung der Servicedaten sowie die Umsetzung der zur Steuerung eines Dienstes nötigen Bedienelemente. Werden semi-automatische oder manuelle Dienste (siehe Kapitel

3.1.3) verwendet, dann erfolgt die Bereitstellung der virtuell kontextbasierten Dienste über ein Angebot der Dienste an den Benutzer. Dieses Dienstangebot muss auch geeignet in der virtuellen Welt dargestellt werden und zusätzliche Bedienelemente für die Aktivierung und Deaktivierung der Dienste müssen bereitgestellt werden. Die Integration der Dienste in die virtuelle Welt ist notwendig für ein ungetrübtes Spielerlebnis. Die Dienstintegration erlaubt eine einheitliche Bedienung, sodass die Spieler gleichzeitig Dienste nutzen und am Spielgeschehen teilnehmen können. Das ist vor allem wichtig, um die Immersion des Spielers nicht zu stören (siehe Kapitel 2). Die Grundlagen für eine Integration von Diensten in virtuelle Welten bilden die Erkenntnisse aus dem Game Design. Für eine Integration in das Spiel müssen die Aktionen und Resultate erkennbar gemacht werden, das heißt sie müssen wahrnehmbar kommuniziert und in den Kontext des Spieles integriert werden [SZ05]. Für die Umsetzung der Integration gibt es verschiedene Möglichkeiten, wie beispielsweise die Integration über das Benutzerinterface der virtuellen Welt, die Integration in andere Interface-Elemente wie Text- oder Voice-Chat, die Integration über eine Visualisierung auf Spielobjekten oder eine Integration mittels der Integration neuer Spielobjekte.

Die Funktionen, die ein System bereitstellen muss, das den Anforderungen genügt, können in vier Bereiche unterteilt werden: um eine situationsabhängige Unterstützung zu ermöglichen, muss der virtuelle **Kontext** des Benutzers betrachtet werden, dem Benutzer müssen innerhalb der virtuellen Welt entsprechende **Dienste** zur Verfügung gestellt werden und der bidirektionale **Austausch der Daten** zwischen virtueller Welt und externen Applikationen muss realisiert werden. Zusätzlich muss ein solches System die benötigten **Verwaltungs**aufgaben übernehmen.

5.1.2 Non-funktionale Anforderungen

Die non-funktionalen Anforderungen, die bei der Realisierung des Informationsaustauschs zwischen Spielen und ihrem Umfeld entstehen, sind *Datensicherheit* sowohl aus Spiel- wie auch aus Benutzersicht und **Benutzerfreundlichkeit**.

Die Kontextinformationen des Benutzers müssen an die verwendeten externen Dienste weitergeleitet werden. Bei der Verteilung der Kontextinformationen an externe Dienste, die diese Kontextinformationen nutzen möchten, muss die Datensicherheit aus Spiel- und Benutzersicht geschützt werden. Der Datenschutz aus Benutzersicht (*Privacy*) beinhaltet, dass der Benutzer über die Weitergabe seiner Kontextinformationen bestimmen können muss. Die spielbezogene Datensicherheit (Schutz vor *Cheating*) bedeutet, dass der Spielbetreiber festlegen kann, wann Kontextinformationen weitergegeben werden dürfen.

Benutzerfreudlichkeit beinhaltet, es der Community zu ermöglichen, eigene Dienste zu entwickeln und bereitzustellen. Dabei ist eine einheitliche Basis für die Entwicklung von Diensten für unterschiedliche virtuelle Welten bereitzustellen, sowie die Unterstützung der Spieler, insbesondere der Gruppeninteraktion und der Community-Entwicklung, innerhalb von virtuellen Welt zu ermöglichen.

Cheating

Die Absicherung der Spiele gegenüber ungewollten Manipulationen, die negative Auswirkungen auf das Spielerlebnis haben und die Gefahren, die durch eine Manipulation der Spiele entstehen können, werden im Folgenden vorgestellt. Bei einer uneingeschränkten Beteiligung der Community, beispielsweise durch *Mods* oder den vollen Zugriff auf in-game Informationen (siehe auch Kapitel 3.3.3), können immer auch solche Veränderungen an Spielen entstehen, die Nachteile für Spieler verursachen oder das Spielen an sich stören oder behindern. Besonders bei Spielen mit vielen Teilnehmern und bei Spielen in denen sportliche Wettkämpfe ausgetragen werden, muss das Spielgeschehen kontrollierbar sein, um unfaire Vorteile ausschließen zu können. Dabei geht es bei der Sicherung der Balance und der Fairness zwischen den Spielern eines Onlinespiels darum, den Spielspaß und gleichzeitig die wirtschaftlichen Interessen der Spieleindustrie zu wahren. Gerade im e-sports Bereich spielt die Fairness und die Einhaltung fester Regeln eine entscheidende Rolle. Mitunter geht es in e-sport Wettkämpfen um sehr viel Geld, sodass auch zur Wahrung der wirtschaftlichen Interessen der involvierten Firmen einheitliche Vorausset-

zungen gewährleistet sein müssen. Aber auch in allen anderen Onlinespielen führt ein übermäßiges oder als unfair empfundenes Ungleichgewicht zu Unmut unter den Spielern und kann dadurch dem Spielspaß und auch dem Spiel erheblich schaden.

In Onlinespielen wird das Verhalten eines Spielers immer dann als *Cheating* bezeichnet, wenn er durch sein Verhalten einen Vorteil gegenüber seinen Mitspielern erhält oder ein Ziel im Spiel erreicht, wobei dieser Vorteil oder dieses Ziel für ihn eigentlich nicht erreichbar sein sollte [YC02] (bezüglich des Regelwerks des Spiels oder des Ermessens des Spielanbieters [YR05]).

Ursprünglich bezeichnete der Begriff Cheating die Ausnutzung von *Bugs* (Fehlern) oder Hintertüren in Spielen. Darüber hinaus haben sich ganz unterschiedliche Formen von Cheating in Onlinespielen entwickelt. Angefangen bei ganz einfachen Formen, wie dem selbst herbeigeführten *Disconnect* (Ziehen des Netzwerksteckers), um ausweglosen Niederlagen im Spiel zu entkommen, über das in der Regel verbotene Handeln mit virtuellen Gegenständen oder Währungen (*Item*-Handel) außerhalb der virtuellen Welt gegen reales Geld oder die Verwendung von unerlaubten Hilfsprogrammen (wie z.B. *Bots* [SGJ07]), bis hin zur direkten Manipulation des Spielesystems (Manipulation das Betriebssystems, Manipulation des Spiels oder Betrug der Mitspieler). Eine Manipulation das Betriebssystems wird beispielsweise beim *wall hack* vorgenommen oder erfolgt direkt auf der Hardware vor allem bei Spielkonsolen. Die Manipulation des Spiels umfasst das Hacking des Spiel-Clients (z.B. durch *maphacks* [CFFS05]) oder des Spiel-Servers. Ein Betrug der Mitspieler erfolgt durch ausspionieren von ID und Passwort oder durch unzulässige Absprachen. Eine systematische Klassifizierung von Cheating in Onlinespielen findet sich in der Taxonomie von Yan und Randell [YR05]. Das Erkennen, Verhindern und die Prävention von Cheating sind heute wichtige Aspekte in der Spieleforschung (siehe auch [FKS08]).

Um Cheating und eine Überschreitung der Regeln, die von Spielerherstellern und Spielanbietern festgelegt werden, zu verhindern, werden sowohl rechtliche als auch technische Maßnahmen ergriffen. Die typische Form der rechtlichen Regelungen in Bezug auf die Verwendung von Spielen sind *End User License Agreements* (EULAs), die die Nutzungsbedingungen eines Spiels definieren.

Zusätzlich werden in Onlinespielen und auf e-sport Veranstaltungen immer neue Mechanismen und technische Vorkehrungen eingesetzt. Dabei wird zum Beispiel *Anti-Cheat* Software (wie PunkBuster [25]), in die Spiele integriert und es werden Methoden entwickelt um sogenannte *Bots* (automatisierte Agenten) von menschlichen Spielern zu unterscheiden (CAPTCHA Tests - *Completely Automated Public Turing Test to Tell Computers and Humans Apart*), um für die Gleichberechtigung aller Beteiligten zu sorgen [GD05]. Die Austragung der meisten professionellen e-sport Wettkämpfe findet bereits auf vom Veranstalter bereitgestellten System und in LAN-Netzwerken statt, nicht nur um kürzere Verzögerungen bei der Übertragung der Daten zu erreichen, sondern auch um das Cheaten unterbinden zu können.

Privacy

Kontext-bewusste Applikationen nutzen den Kontext eines Benutzers, um ihre Dienste und die Informationen, die dem Benutzer bereitgestellt werden, anzupassen ([KPM05]). Dazu werden diese Dienste über Applikationen bereitgestellt, die über alle Kontextänderungen, über die sie informiert werden wollen, informiert werden ([CK00]). Eine Möglichkeit für den Benutzer, seine Kontextinformationen zu schützen, besteht zum Beispiel in der vorherigen Definition von Richtlinien für die Kontextverwendung. Diese sind jedoch statisch und bieten nur die Möglichkeit, vertrauenswürdige Applikationen zu definieren (unabhängig von den genutzten Kontextinformationen) [CFJ04]. Zu ermöglichen, dass ein Benutzer innerhalb einer virtuellen Umgebung Zusatzdienste verwenden kann, die durch verteilte Internet-Applikationen bereitgestellt werden, und dabei die Kontrolle über die Verwendung seiner personenbezogenen Kontextinformationen behält, erzeugt spezielle Anforderungen an Verarbeitung und Weitergabe der zur Verfügung stehenden Kontextinformationen. Dabei ist entscheidend, dass zum einen Dienste nur in einem vorher definierten Kontext aktiv sind, also nur bei aktiven Diensten Kontextinformationen weitergegeben werden, und dass der Benutzer vor der Verwendung eines Dienstes, bevor Kontextinformationen weitergegeben werden, der Dienstnutzung zustimmen muss. Kontextbasierte Dienste werden situationsbedingt einem Benutzer angeboten und passen sich dann bei Verwendung dem aktuellen Kon-

text an. Dabei wird der Benutzer vor der Verwendung eines Dienstes über die weitergegebenen Kontextinformationen informiert. Applikationen, die kontextbasierte Dienste bereitstellen, bekommen nur dann Kontextinformationen mitgeteilt, wenn ein Benutzer einen entsprechenden Dienst auch verwendet.

Benutzerfreundlichkeit

Die Tools, die im Rahmen von Community-Aktivitäten angeboten werden, stellen für die aktive Beteiligung der Spieler an den verschiedenen Aktivitäten oft selbst eine Herausforderung dar. Es werden verschiedenartige Tools, wie Level-Editoren, PHP-Anwendungen, Kollaborations-, Koordinations- oder Kommunikationstools, sowie verschiedene Zugangsmöglichkeiten Skriptsprachen, Schnittstellen, etc. verwendet. Beispielsweise werden als Modding-Tools teilweise sogar die Programme bereitgestellt, die auch die Spieleentwickler selbst verwenden. Das erlaubt eine sehr große Flexibilität, schränkt auf der anderen Seite jedoch den möglichen Benutzerkreis ein, da spezielle Kenntnisse für die Verwendung dieser Programme benötigt werden. Das größte Problem entsteht dabei durch die Verschiedenartigkeit der Programme und Systeme. Da es keinen einheitlichen Standard gibt, existieren von Spiel zu Spiel unterschiedliche Technologien und abweichende Verwendungsarten entwickeln sich. Dadurch üben die Communities verschiedener Spiele zwar ähnliche Tätigkeiten aus, ihnen liegen dabei jedoch größtenteils unterschiedliche Schnittstellen, Programmier- und Skriptsprachen oder gar unterschiedliche Plattformen zugrunde. Für engagierte Spieler, die parallel in den Communities verschiedener Spiele aktiv sind oder von einem Spiel zu einem anderen wechseln, bedeuten die Kompatibilitätsprobleme einen Mehraufwand, da der Spieler gleichartige Tätigkeiten oft unterschiedlich angehen muss und hierfür zunächst neue Kenntnisse und Fähigkeiten erwerben muss.

Die Unterstützung der Spieler und der Community innerhalb der virtuellen Welt ist eine weitere Herausforderung. Gerade für MMORPGs, in denen das Bilden von Gruppen nicht nur durch das Gameplay motiviert sondern auch gefordert ist, finden sich keine ausreichenden Mechanismen, diese Gruppen adäquat zu unterstützen. Eine Unterstützung innerhalb des Spieles ist jedoch unabdingbar für größere Gruppen. Teilweise bieten MMOGs innerhalb des Spiels eine Unterstützung für die Organisation der Spieler in Clans oder Gilden an. Diese Lösungen decken jedoch nur einen geringen Teil des Bedarfs der Communtities ab. Um eine Gruppe von Spielern zu organisieren, sind zusätzliche Verwaltungsmöglichkeiten nötig (z.B. für das Finden neuer Mitglieder, die Koordination von Events oder um Aufgaben zu verteilen). Ducheneaut et al. [DYNM07] haben in ihrer Untersuchung von Gilden in Online-Games festgestellt, das Gruppen ab 35 Mitgliedern ohne zusätzlich Unterstützung nicht aufrecht zu erhalten sind und das schon der Zugang zu Informationen über die Gilden eine große Verbesserung bringt. Wenn ein Spieler beispielsweise seine Spielerrungenschaften auf einer Webseite präsentieren kann, dann kann er dadurch seinen Status anderen Spielern in der Community zeigen. Somit wird die Visibilität erhöht und die Suche nach passenden Teammitgliedern vereinfacht.

5.2 Bestehende Ansätze

Die aus den entstehenden sozialen Vernetzungen resultierenden Aktionen der Spieler und die Nutzung von Applikationen im Umfeld eines Spiels erzeugen den Bedarf einer Vernetzung des Spielumfelds und der stattfindenden Community-Aktivitäten mit dem zugrundeliegenden Spiel. Die technischen Aspekte einer Vernetzung von Spielen mit ihrem Spielumfeld spielen jedoch in der Forschung noch keine Rolle, im Gegensatz zu den sozialen Aspekten der Vernetzung, die sowohl im Spiel als auch darüber hinaus im Spielumfeld betrachtet werden (siehe 3.3.3). Jedoch gibt es bereits einzelne Ansätze, die den Zugriff auf in-game Informationen ermöglichen können. Für die Realisierung kontextbasierter Dienste gibt es verwandte Ansätze im Bereich der kontext-bewussten Dienste (siehe Kapitel 3.1.4). Es folgt die Betrachtung der bestehenden Ansätze zum Informationsaustausch und deren Beurteilung, sowie die Beurteilung der verwandten Ansätze aus dem Bereich der kontext-bewusster Dienste.

5.2.1 Informationsaustausch zwischen Spielen und anderen Internet-Applikationen

Um eine Verknüpfung zwischen Spiel und Spielumfeld zu erreichen, wurden erste Ansätze entwickelt, die einen Zugriff auf in-game Informationen ermöglichen. Dabei kann man drei unterschiedliche Herangehensweisen unterscheiden. Diese sind *entwicklerinitiierte Spieledatenpräsentation*, *Launch Plattformen* und *Webportale*.

Entwicklerinitiierte Spieledatenpräsentation

Bei der entwicklerinitiierte Spieledatenpräsentation werden in-game Informationen von den Entwicklern oder Publishern des Spiels extern zur Verfügung gestellt. In den meisten Fällen verwendet der Spielanbieter dazu spezielle Webseiten. Auf diesen Webseiten werden bestimmte Informationen aus einem Spiel und über seine Spieler angezeigt. Die Entwickler verwenden dafür eine unidirektionale Verbindung vom Spieleserver zur Webseite (siehe Abbildung 5.3).

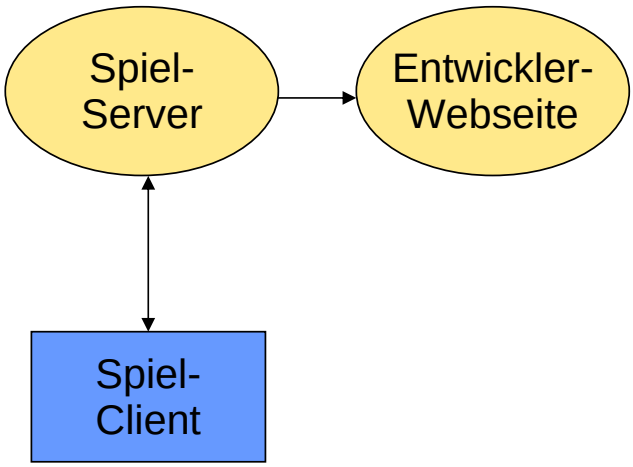

Abbildung 5.3: Datenfluss bei entwicklerinitiierter Spieledatenpräsentation

Es gibt auch andere Formen der entwicklerinitiierte Spieledatenpräsentation. Diese sind jedoch wenig verbreitet oder auf sehr spezielle Daten beschränkt. Ein Beispiel, in dem ein ganz anderer Kommunikationsmechanismus verwendet wird, ist die SMS-Benachrichtigung. Diese wird beispielsweise in Browsergames eingesetzt (*Mafia - Gangs of Crime 1930* [6] oder *SchoolWars* [59]). Der Benutzer bekommt dabei eine SMS-Benachrichtung zugeschickt, die ihn über spezielle Ereignisse im Spiel informiert, etwa darüber, dass er im Spiel angegriffen wird. Die Kosten für die zugesendete SMS muss der Benutzer normalerweise selbst tragen.

Die entwicklerinitiierte Spieledatenpräsentation auf speziellen Webseiten bietet eine verläßliche Sammlung von Daten über Gegenstände und / oder Charaktere eines Spiels. Ein Spieler hat auf die dargestellten Informationen jedoch keinen Einfluss und kann auch selbst keine eigenen Information hinzufügen. Dafür können die präsentierten Daten teilweise von anderen Webseiten weiterverwendet werden. Beispiele für entwicklerinitiierte Spieledatenpräsentation auf Webseiten sind die Statistik-Webseite von *Enemy Territory: Quake Wars* [2] und die *World of Warcraft* Armory ("Das Arsenal" [7]) siehe Abbildung 5.4.

Entwickler-Seiten von FPS oder RTS Spielen enthalten meist spielerbezogene oder sessionbezogene Daten. Auch die **Statistik-Webseite** *Enemy Territory: Quake Wars* beinhaltet solche Spielerstatistiken, wie die Benutzung verschiedenen Klassen oder Waffen, *win/loss* Raten oder *Achievements* (Errungenschaften) der Spieler.

Webseiten zu MMOGs enthalten vor allem Informationen über den aktuellen Zustand der Spielercharaktere oder Informationen zur Spielwelt, wie auch die *World of Warcraft* **Armory**, die Informationen

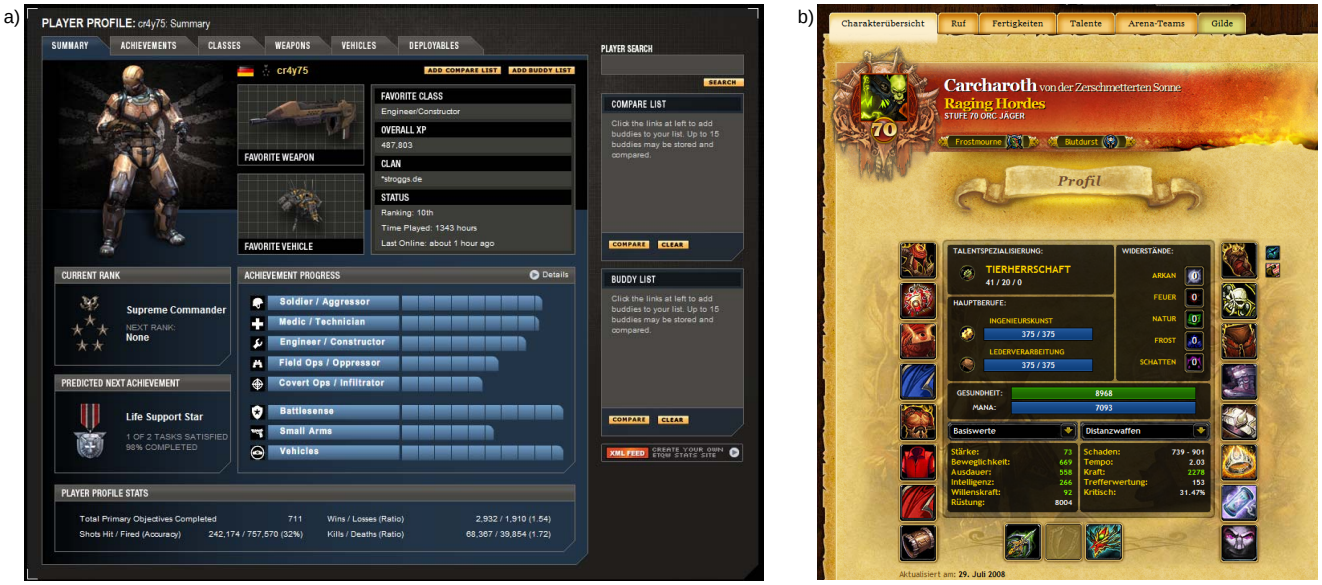

Abbildung 5.4: Beispiele für entwicklerinitiierte Spieledatenpräsentation: a) Statistik-Webseite von *Enemy Territory: Quake Wars* [2] b) *World of Warcraft* Armory [7])

über Gildenzugehörigkeit, Ausrüstung oder Skills eines Charakters enthält aber auch Gegenstände und NPCs der Spielwelt verzeichnet.

Launch Plattformen

Launch Plattformen sind eine Sammlung verschiedener kleiner Applikationen. Dabei ist nicht festgelegt was genau eine Launch Plattform enthalten muss oder kann. Typischweise beinhalten Launch Plattformen *Server-Browser* (eine Applikation zum Auffinden geeigneten Server bei FPS oder RTS Spielen), Kommunikationssoftware (zum Beispiel *Instant Messenger*, IM) und *Buddylisten* (um Freunde zu organisieren). Die Launch Plattform ihrerseits muss auf dem Client-System des Benutzers installiert werden und tauscht von da aus Informationen mit den angebundenen Internet-Applikationen aus. Diese Plattformen verwenden teilweise eine Überwachung der Prozesse bestimmter Spiele, sodass die Launch Plattform Kenntnis darüber hat, welches Spiel gerade gespielt wird und diese Informationen an die Freunde des Spieler weitergeben kann. Somit besteht eine eingeschränkte unidirektionale Verbindung vom Spiel-Client zur Launch Plattform (siehe Abbildung 5.5).

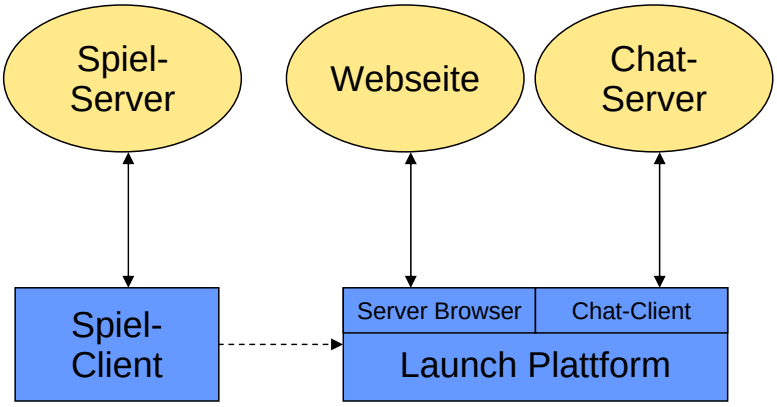

Abbildung 5.5: Datenfluss bei einer Launch Plattform

Manche Launch Plattformen bieten dem Benutzer zusätzlich zu der Kommunikation über die Launch Plattform ein grafisches *Messaging-Overlay*, das ein Kommunikationsfenster öffnet welches über anderen

Applikationen (auch Spielen) dargestellt wird und somit ohne ein Verlassen des Spiels verwendet werden kann. Beispiele für Launch Plattformen sind ESLwire [23], Steam [62] oder Xfire [67].

ESLwire ist die Launch Plattform der ESL (Electronic Sports League [61]) und gekoppelt mit den ESL-Benutzerprofilen. Sie unterstützt fast alle Spiele, die in der ESL gespielt werden, bietet Buddylisten und Kommunikationstools (IM und Gruppen-Chat) und enthält *Game-Lobbies* für das Finden von Mitspielern.

Steam ist die Launch Plattform von Valve [63] und Voraussetzung für das Spielen aller Steam-Spiele. Sie enthält neben einem Messenger, Buddylisten und Komminikationstools (Chat), ein Overlay für die Steam-Community.

Xfire ist in erster Linie ein Instant Messenger. Er erlaubt zusätzlich den Austausch von Dateien und Screenshots zwischen Freunden. Der integrierte Server-Browser kann über 1000 Spiele erkennen und ein Message-Overlay für die Kommunikation ist ebenfalls enthalten.

Webportale

Im Gegensatz zu entwicklerinitiierter Spieledatenpräsentation werden Webportale von Drittanbietern gehostet. Dadurch haben Webportale keinen direkten Zugriff auf Daten von den Spiele-Servern. Webportale unterstützen in der Regel nur eine kleine Auswahl an Spielen oder sind ganz auf ein Spiel spezialisiert. Dafür nutzen sie, wenn möglich, die Daten, die die Publisher auf ihren Webseiten bereitstellen (beispielsweise aus der *World of Warcraft* Armory) und stellen auch manuell zusammengetragene Informationen und News über die Spiele bereit. Manche Webportale verwenden zusätzlich Client-Applikationen, die über spezielle Spiele-Addons Daten über den Benutzer erfassen und nach Beendigung des Spiels an das Webportal weitergeben (siehe Abbildung 5.6).

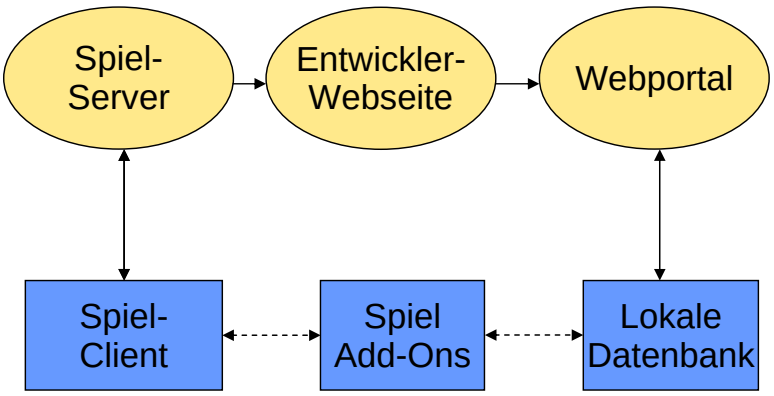

Abbildung 5.6: Datenfluss bei einem Webportal

Der Schwerpunkt der Webportale liegt auf der Eigenpräsentation der Spieler. Dafür stellen sie soziale Netzwerke (mit Freundeslisten, Diskussionsgruppen, etc.) bereit und geben den Spielern Raum, um sich selbst darstellen zu können (zum Beispiel über Benutzerprofile in Blogs). Darüber hinaus haben Webportale wie buffed.de [17], Xchar [80] oder rupture [22] begonnen, verfügbare in-game Informationen aufzubereiten und wiederum der Community bereitzustellen.

Buffed.de wurde ursprünglich als Webportal für *World of Warcraft* entwickelt unterstützt jedoch inzwischen auch ein paar weitere Spiele. Die *my*buffed Community-Features bieten ein soziales Netwerk und Blogs mit Profilen und virtuellen Charakteren. Buffed.de bietet zusätzlich den BLASC-Client an, der ein *World of Warcraft* Addon enthält, über das Daten des Spielers gesammelt werden können.

Xchar ist ein Webportal rein für *World of Warcraft* Spieler. Neben einem sozialen Netzwerk bietet Xchar ein spezielles *World of Warcraft* Addon an, das Angaben aus dem Xchar Profil im Spiel verlinkt, um dadurch eine Verknüpfung der Beziehungen von Spielern innerhalb und außerhalb des Spiels zu unterstützen. Auch die Möglichkeit, eine spielbezogene Signatur zu erstellen, wird angeboten.

Rupture befindet sich noch in der Entwicklung (Beta-Phase). Es unterstützt verschiedene Spiele und hat die Möglichkeit, Daten von anderen Webseiten (entwicklerinitiierte Spieledatenpräsentation, wie

World of Warcraft Armory oder Xbox Live) wiederzuverwenden. Die Entwicklung von Rupture ging schon in verschiedene Richtungen nachdem es von Electronic Arts (EA) [20] übernommen wurde ist noch offen welche Richtung die weitere Entwicklung von Rupture nehmen wird.

Die vorgestellten Ansätze betrachten die Übertragung von Information aus einem Spiel in dessen Spielumfeld. Der entstehende Informationsfluss vom Spiel zu den verschiedenen Internet-Applikationen ist in den meisten Fällen auf einzelne Spiele und bestimmte Applikationen beschränkt, sowie in der Regel nur eingeschränkt verfügbar.

Für den Informationsfluss vom Spielumfeld in das Spiel hinein gibt es bisher nur vereinzelte spielspezifische Ansätze, wie zum Beispiel die Möglichkeit spezialisierte Addons zu entwickeln. Diese Möglichkeiten werden jedoch nur vereinzelt von Spielen bereitgestellt und erlauben in der Regel nur einen asynchronen Informationsaustausch (offline Datenaustausch).

5.2.2 Realisierung kontext-bewusster Dienste

Viele der in der Literatur zu kontext-bewussten Systemen (siehe auch Abschnitt 3.1) diskutierten Themen und Fragestellungen sind auch relevant in einem System für die Bereitstellung von externen Diensten in virtuellen Welten. Bei kontext-bewussten Systemen werden die Verwendung von Diensten in Verbindung mit Kontextinformationen, wie auch andere verwandte Aspekte (wie Erfassung, Sammlung und Verteilung von Kontextinformationen), betrachtet.

Auch für die Umsetzung einer auf Kontextinformationen basierenden Dienstarchitektur, gibt es verwandte Arbeiten aus dem Bereich der kontext-bewussten Systeme. Verschiedene Middleware-Architekturen kontext-bewusster Systeme wurden in Abschnitt 3.1.4 vorgestellt. Mit einer Middleware, wie sie im Bereich der kontext-bewussten Systeme eingesetzt wird, beispielsweise basierend auf dem "Context Toolkit Framework", kann jedoch nur ein Teil der Anforderungen erfüllt werden. So kann die Erfassung des virtuellen Kontexts über Software-Sensoren (Spiele-APIs), die Kapselung der Kontextakquise wie in Widgets und eine benutzerbezogene Aggregation der Informationen durchgeführt werden, wobei externe Applikationen diese Informationen dann verwenden können.

Die Architektur eines kontext-bewussten Systems wird jedoch vor allem durch die Methode der Kontextakquise bestimmt [BDR07]. Der Hauptgrund für die Verwendung einer solchen Architektur ist die Trennung der verschiedenen Aspekte und Sichtweisen, auf der einen Seite die Kontexterfassung über Sensoren und auf der anderen Seite die Kontextnutzung.

5.2.3 Beurteilung bestehender Ansätze

Im Folgenden findet sich eine Beurteilung der vorgestellten Ansätze bezüglich des Szenarios und der Zielsetzung dieser Arbeit.

Informationsaustausch

Allen bestehenden Ansätzen aus dem Spielebereich ist gemein, dass sie nicht generisch, d.h. für verschiedene Spiele einerseits und verschiedene Anwendungstypen in der Spielumgebung andererseits nutzbar sind. Die größten Nachteile der bestehenden Ansätze liegen in Unidirektionalität und Asynchronität des Informationsaustauschs und in der durch die Spezialisierung beschränkten Funktionalität. Es gibt keinen direkten Informationsaustausch während sich der Spieler im Spiel aufhält und es werden meist nur aggregierte oder zwischengespeicherte Daten weitergegeben. Die Funktionalität ist eingeschränkt und erlaubt entweder nur eine Datenpräsentation und keine Interaktionsmöglichkeiten, wie bei den Entwickler-initierte Spieledatenpräsentation, oder sie ist beschränkt auf Tools zur Interaktionsunterstützung, wie die Kommunikationstools, Buddylisten oder Server Browser der Launch Plattformen oder wie

die sozialen Netzwerke der Webportale. Den umfangreichsten Zugriff auf ingame-Informationen bieten Ansätze von Spielentwicklern und Publishern. Von diesen ist jedoch keine einheitliche generische Lösung zu erwarten, da sie ihre eigenen Ansätze als Wettbewerbsvorteil sehen, den sie nicht mit der Konkurrenz teilen werden.

Es gibt noch keinen Ansatz, der eine generische Lösung für den Informationsaustausch zwischen Spielen und den unterschiedlichen Applikationen im Spielumfeld, sowie eine in-game Unterstützung der Spieler ermöglichen kann.

Kontext-bewusste Dienste

Die Hauptunterschiede zwischen kontext-bewussten Systemen und kontextbasierten Diensten bestehen in den verwendeten Kontextinformationen beziehungsweise der unterschiedlichen Quelle des Kontexts (real / virtuell), in der Art und Weise der Verwendung von Kontextinformationen (kontext-bewusst / kontextbasiert) und in der Einbettung der Dienste in das Gesamtsystem. Dabei unterscheiden sich realer und virtueller Kontext bezüglich des zugrundeliegenden Systems, in dem die Kontextinformationen vorliegen. Anstelle einer (realen) Welt gibt es eine Menge an unterschiedlichen virtuellen Umgebungen, die berücksichtigt werden müssen. Dadurch variiert nicht nur die Kontextakquise sondern vor allem auch die Verarbeitung der Kontextinformationen.

Wie schon in 4.4.1 dargestellt, können Konzepte aus dem Bereich der kontext-bewussten Dienste in Ergänzung zu kontextbasierten Diensten eingesetzt werden und es gibt auch verschiedene Parallelen zwischen beiden Bereichen. Die für das Szenario dieser Arbeit identifizierten Anforderungen an ein unterstützendes System sind im Vergleich zu den Anwendungsbereichen in denen kontext-bewusste Systeme eingesetzt werden (wie *smart spaces*) zu unterschiedlich, als dass eine Anpassung eines kontext-bewussten Systems für die Umsetzung einer Verknüpfung von Spielen und Internet-Applikationen in ihrem Umfeld sinnvoll ist (siehe 3.1.4).

Fazit

Um den Anforderungen gerecht zu werden, muss eine generische Lösung geschaffen werden, die zum einen den Informationsaustausch zwischen einem Spiel und anderen Internet-Applikationen im Spielumfeld und zum anderen die Realisierung kontextbasierter Dienste in virtuellen Welten ermöglicht, sowie die Community aktiv einbindet, eine Unterstützung innerhalb der virtuellen Welt erlaubt und negativen Auswirkungen auf das Spiel vorbeugen kann. Dabei müssen sowohl in-game Aktivitäten oder Informationen im Spielumfeld zugreifbar gemacht, sowie Daten und Community-Aktivitäten aus dem Spielumfeld ins Spiel integriert werden. Ein generischer Ansatz für die Verknüpfung von Spiel und Spielumfeld kann ganz neue Möglichkeiten für das Spiel und die kreative Beteiligung der Community eröffnen. Im Folgenden wird die in dieser Arbeit entwickelte generische Lösung vorgestellt, die einen bidirektionalen Informationsaustausch basierend auf virtuell kontextbasierten Diensten darstellt und die bestehenden Herausforderungen erfüllt.

5.3 VCBS-Middleware - Design und Eigenschaften

Eine generische Lösung zur Realisierung des Informationsaustauschs zwischen Spielen und ihrem Umfeld sowie von virtuell kontextbasierten Diensten bietet die in dieser Arbeit entwickelte VCBS-Middleware. Die VCBS-Middleware ist ein neuartiger Ansatz, um einen standardisierten Informationsaustausch zwischen Spiel und Spielumfeld zu ermöglichen. Mit der VCBS-Middleware wird das Konzept der virtuell kontextbasierten Dienste umgesetzt, deren Ziel es ist, dem Benutzer, passend zu seiner aktuellen virtuellen Situation, Dienste bereitzustellen (siehe auch Abschnitt 4.4.1).

Die VCBS-Middleware realisiert die Verwendung von semi-automatischen Diensten (siehe Abschnitt 3.1.3). Der Benutzer wählt dabei die Dienste aus, die er verwenden möchte, und erst danach werden diese aktiviert und erhalten die entsprechenden Kontextinformationen. Zusätzlich sind die Kontextinfor-

mationen, welche von der VCBS-Middleware an die externen Dienste weitergegeben werden, durch die Regeln der Spielanbieter eingeschränkt, um deren missbräuchliche Verwendung zu verhindern. Andere Formen von Diensten (wie automatische Dienste) sind mit virtuellem Kontext auch realisierbar, widersprechen jedoch dem Prinzip der VCBS-Middleware, die den Benutzer bewusst aktiv in die Dienstauswahl einbindet, um eine situations- und benutzerspezifische Unterstützung umzusetzen.

Im Folgenden wird die Funktionsweise der VCBS-Middleware erläutert und auf spezielle Aspekte eingegangen. Dafür werden konzeptionelle Aspekte (wie kann man externe Dienste anbinden) und technische Aspekte (wie kann man eine solche Anbindung umsetzten) betrachtet.

5.3.1 Funktionsbereiche der VCBS-Middleware

Die VCBS-Middleware setzt die vier identifizierten Funktionsbereiche (siehe Abschnitt 5.1.1) **Kontext**, **Dienste**, **Daten** und **Verwaltung** um. Abbildung 5.7 stellt die Funktionskomponenten der Middleware dar und zeigt die Vernetzung der einzelnen Komponenten.

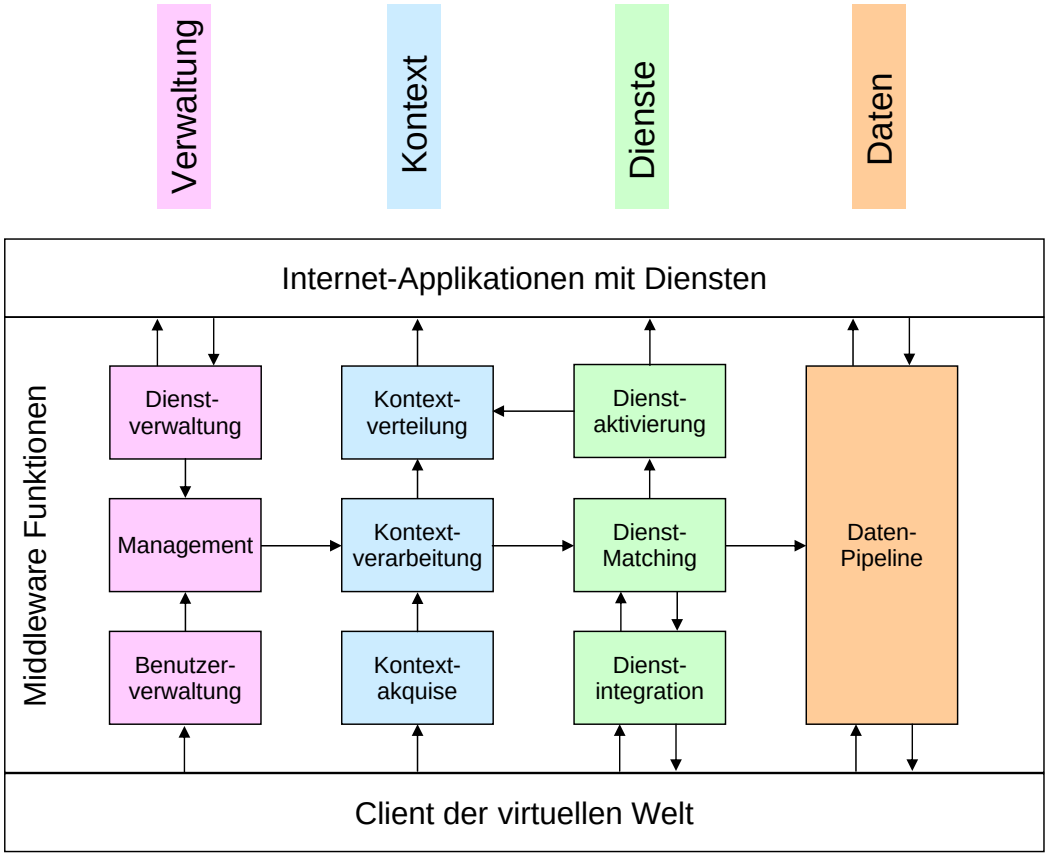

Abbildung 5.7: Funktionen der Middleware

Im Funktionsbereich *Kontext* werden durch die VCBS-Middleware die virtuellen Kontextinformationen des Benutzers vom Client einer virtuellen Welt erfasst (*Kontextaquise*). Die *Kontextverarbeitung* ermittelt aus den erfassten Kontextinformationen die virtuelle Situation des Benutzers, welche für das kontextbasierte Angebot von Diensten und deren kontextbasierte Bereitstellung benötigt wird. Die Informationen werden zum einen den externen Diensten bereitgestellt (*Kontextverteilung*). Zum anderen werden die Informationen über die virtuelle Situation an das *Dienst-Matching* im Funktionsbereich *Dienste* weitergegeben.

Um eine Bereitstellung von externen Diensten innerhalb der virtuellen Welt zu ermöglichen, übernimmt die VCBS-Middleware die *Dienstintegration*. Die Integration umfasst das Dienstangebot, sowie

die Benutzerinteraktion mit den externen Diensten. Im Rahmen der *Dienstintegration* werden die aktuell verfügbaren Dienste dem Benutzer über den Client der virtuellen Welt angeboten und die vom Benutzer vorgenommene Aktivierung und Deaktivierung der angebotenen Dienste erfasst. Wenn ein Benutzer einen Dienst aktiviert oder deaktiviert wird dies der *Dienst-Matching* Komponente mitgeteilt. Dafür informiert die *Dienst-Matching* Komponente über die in der aktuellen Situation verfügbaren Dienste. Das *Dienst Matching* ist ein kontextbasierter Dienststeuerungs-Mechanismus, der für den Benutzer ein situationsabhängiges Dienstangebot erstellt und basierend auf der Benutzerinteraktion die *Dienstaktivierung* übernimmt, wenn der Benutzer einen Dienst verwenden möchte. Diese Information wird der *Dienstaktivierungs*-Komponente mitgeteilt. Diese teilt zum einen der Internet-Applikation, die den externen Dienst bereitstellt, mit, dass der Benutzer den Dienst verwendet. Zum anderen informiert die *Dienstaktivierungs*-Komponente die *Kontextverteilung*, die nun die entsprechenden Kontextinformationen des Benutzers an den externen Dienst weitergibt. Dadurch kann die Kontextverteilung benutzer- und dienstspezifisch vorgenommen werden.

Bei der Nutzung von Diensten übernimmt die VCBS-Middleware die Integration der Servicedaten in die virtuelle Welt. Diese werden dafür von der VCBS-Middleware vom Dienst über die *Daten-Pipeline* zum Benutzer übertragen. Diese übernimmt auch die Übertragung weiterer Informationen aus dem Client der virtuellen Welt an den externen Dienst, wie beispielsweise Benutzereingaben. Der über die *Daten-Pipeline* realisierte Informationsaustausch wird nur für aktive Dienste durchgeführt. Die Informationen über den Zustand der Dienste erhält die *Daten-Pipeline* von der *Dienst-Matching*-Komponente.

Der Bereich *Verwaltung* umfasst das *Management* der Middleware, sowie die Verwaltung der Benutzer und der Dienste. Funktionen der *Benutzerverwaltung* sind Benutzerregistrierung und Authentifizierung. Für die Identifikation eines Benutzers werden Informationen über seinen virtuellen Charakter verwendet. Informationen über die An- und Abmeldung der Benutzer werden an die *Management*-Komponente weitergegeben. Bei der *Dienstverwaltung* der VCBS-Middleware können Dienste registriert werden. Nur Dienste, die auch regelkonform bezüglich eines Regelsatzes, den der Spieleanbieter definieren kann sind, werden von der VCBS-Middleware angeboten. Die Regelwerke sind Teil der *Dienstverwaltung* und haben Einfluss auf die Verwendung der Kontextinformationen durch die externen Dienste. Die Informationen über registrierte Dienste werden an das *Management* weitergegeben. Im *Management* der Middleware laufen alle zur Verwaltung der Middleware benötigten Informationen zusammen und die zur Verarbeitung der Kontextinformationen der Benutzer benötigten Informationen werden von dort an die *Kontextverarbeitung* weitergegeben.

Das vorgestellte Design der Middleware hat verschiedene Vorteile. Die VCBS-Middleware kann zum einen die Daten der verschiedenen Dienste kanalisieren, sodass der Benutzer nur eine lokale Applikation für die Verwendung der Dienste benötigt und der Client der virtuellen Welt nicht durch verschiedene Applikationen angefragt wird. Dadurch wird auch der entstehende Netzwerk-*Traffic* gebündelt. Es wird nur ein Kommunikationskanal zur Middleware benötigt und es muß nicht zu jedem Dienst einzeln eine Verbindung aufgebaut werden (was im schlimmsten Fall die Netzwerkkommunikation des Spiels beeinträchtigt). Neben der Verbesserung der Kommunikation ist der zweite wichtige Vorteil die Kontrolle des Informationsflusses. Zum einen werden die Informationen, die von einer externen Applikation angefragt werden, auf ihre Zulässigkeit überprüft und zum anderen werden nur Kontextinformationen und Daten an vom Benutzer aktivierte Dienste weitergeleitet. Darüber hinaus erhält der Benutzer auch nur Servicedaten von Diensten, die er aktuell verwendet. Ermöglicht man jedem Dienst den uneingeschränkten Zugriff auf die Kontextinformationen, kann dies eine Verletzung der Privacy des Benutzers bedeuten und es besteht die Möglichkeit, dass die Information missbräuchlich benutzt werden (Cheating).

5.3.2 Aufbau der VCBS-Middleware

Die VCBS-Middleware ist unterteilt in eine *Client-Komponente*, die sich auf dem System des Benutzers befindet, und eine *Online-Komponente*. Abbildung 5.8 zeigt schematisch den Informationsfluss zwi-

schen einem Client-Server *Spiel* und verschiedenen über die VCBS-Middleware angebundenen Internet-Applikationen wie *Community-Portale*, *Wikis* oder *Voice-Chats*.

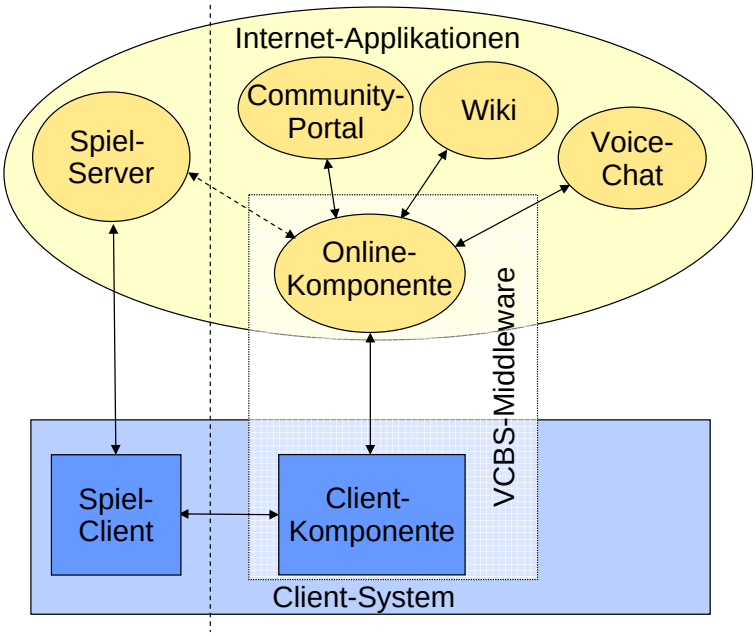

Abbildung 5.8: Informationsfluss der Verknüpfung mit der VCBS-Middleware

Die Unterteilung in diese beiden Komponenten resultiert daraus, dass sich der Spiel-Client als eines der zu verknüpfenden Elemente auf dem Client-System des Benutzer und die Internet-Applikationen als zweites Element im Internet befindet.

Eine integrierte Erweiterung der Funktionalität einer virtuellen Welt durch zusätzliche Dienste und die damit verbundene Erhöhung des Kommunikationsaufkommens zwischen Client und Server der virtuellen Welt, hat, sofern diese bisher realisiert wurde, zu Latenzproblemen und zur Beeinträchtigung des Spiels geführt (wie z.B. bei der Integration eines Voice-Chats in das Spiel *World of Warcraft*). Die Trennung der Übertragung von Spieledaten und Servicedaten ermöglicht eine Entkopplung und darüber hinaus eine Priorisierung der Spieledaten, sodass Zusatzdienste, wenn benötigt, Bandbreite zugunsten des Spieles abgeben können.

Die **Client-Komponente** der VCBS-Middleware auf dem System des Benutzers definiert ein Interface zum Client der virtuellen Welt. Über das Interface können virtuelle Kontextinformation erfasst sowie Servicedaten in-game bereitgestellt werden. Die Client-Komponente der VCBS-Middleware verwendet spielspezifische Plug-Ins, die bestehende Spiel-APIs nutzen, um den benötigten Informationsaustausch zu realisieren. Dabei muss die Einschränkung berücksichtigt werden, dass nur solche Informationen, die durch die API bereitgestellt werden, für die Middleware zur Verfügung stehen. Daher können nicht in jedem Fall alle virtuellen Kontextinformation erfasst werden. Wünschenswert ist, dass die Spielehersteller selbst, basierend auf einer Standardisierung der virtuellen Parameter, die Informationen dem Interface direkt zur Verfügung stellen. Zum Zeitpunkt dieser Arbeit gibt es jedoch noch kein solches Spiel.

Die **Online-Komponente** ist für die Verarbeitung der Kontextinformationen und Servicedaten zuständig und definiert ein Interface für die externen Dienste. Die Online-Komponente verlagert zum einen die Datenverarbeitung vom System des Benutzers weg, wodurch die Ressourcen des Benutzersystems nicht beansprucht werden (und somit dem Spiel zur Verfügung stehen) und ermöglicht zum anderen eine unabhängige Kommunikation der Middleware mit den externen Dienstanbietern, die die Kommunikation des Benutzers nicht beeinträchtigt.

Der Informationsaustausch zwischen der Client-Komponente und der Online-Komponente der VCBS-Middleware weist einige Gemeinsamkeiten zum Informationsaustausch innerhalb von MOGs (zwischen

Spiel-Client und Spiel-Server) auf. So werden zum einen ähnlich zu den Updates eines Spiel-Clients von der Client-Komponente regelmäßig Kontextinformationen an die Online-Komponente übertragen. Dabei ist die Datenmenge eher klein, jedoch müssen die Informationen häufig gesendet werden, um auf Kontextänderungen reagieren zu können. Zum anderen kommen, ähnlich zu den Updates eines Spiel-Servers an den Spiel-Client, von der Online-Komponente regelmäßig Updates mit den aktuellen Dienstangeboten, wobei die übertragene Datenmenge dabei ebenfalls sehr klein ist. Zusätzlich entsteht, durch die Übertragung der Servicedaten, ein weiterer Datenstrom von der Online-Komponente zur Client-Komponente, mit variablen Datenmengen.

Der Aufbau der Online-Komponente der VCBS-Middleware orientiert sich an den Erkenntnissen der Vernetzung von MOGs (siehe Abschnitt 3.2.3). Auch hier gilt es, zwischen verschiedenen Aspekten abzuwägen. Dabei spielen Skalierbarkeit und Sicherheit eine entscheidende Rolle. Der Unterschied zwischen der Online-Komponente der Middleware und dem Server eines MOGs ist, dass ein Spiel-Server einen konsistenten Zustand der Spielwelt benötigt, um die Interaktionen zwischen den Benutzern auswerten zu können (siehe beispielsweise [PW02b] oder [SKH02a]). Die VCBS-Middleware arbeitet benutzerzentriert, sie betrachtet die virtuellen Situationen der einzelnen Benutzer. Dadurch benötigt sie kein globales Wissen über den Zustand des Spiels. Für jeden Benutzer wird ein eigener virtueller Zustand bestimmt, um auf ihn angepasste Dienste anbieten zu können. Zur Realisierung der VCBS-Middleware wird eine verteilte Client-Server Architektur eingesetzt. Die Benutzer können dabei rein nach Netzwerk- und Last-Aspekten auf verschiedene Server verteilt werden. Eine P2P-Architektur erweist sich, aufgrund der Schwierigkeit, Sicherheit (vor allem im Bezug auf den Schutz der Daten) in einem verteilten System zu gewährleisten (siehe Abschnitt 3.2.3), auch hier als ungeeignet.

Die Online-Komponente realisiert die Mehrzahl der zuvor benannten Funktionsbereiche der VCBS-Middleware. Sie besteht aus einer zentralen Verwaltungseinheit für die Benutzer der Middleware und die durch die Middleware angebundenen externen Dienste. Neben der Verwaltungseinheit beinhaltet die Online-Komponente Verarbeitungseinheiten für die Verarbeitung von Kontextinformationen, Daten und Diensten. Dabei ist die VCBS-Middleware nicht auf ein Rechnersystem beschränkt, sondern besteht aus mehreren verteilten Servern unter denen die entstehende Last verteilt wird (*load balancing* / Skalierbarkeit) und die eine gewisse Robustheit des Systems gegen Ausfall einzelner Server darstellen. Die Verwaltungseinheit hat im Vergleich zu den Verarbeitungseinheiten ein vernachlässigbares Kommunikationsaufkommen, da nur Registierung, An- und Abmeldung darüber abgewickelt werden. Der größte Bereich der Verwaltungseinheit besteht in der Verwaltung dieser Prozesse sowie in der Vorhaltung der internen Daten (über Benutzer und Dienste).

5.3.3 Anbindung externer Dienste und deren Integration in virtuelle Welten

Das Ziel der VCBS-Middleware ist die Anbindung und Integration externer Dienste in virtuelle Welten. Dazu realisiert die VCBS-Middleware einen Informationsaustausch zwischen dem Client der virtuellen Welt und den von verschiedenen Internet-Applikationen bereitgestellten externen Diensten. Die VCBS-Middleware ermöglicht die Integration von Diensten und den Austausch virtueller Kontextinformationen und externer Daten, sodass verschiedenste Applikationen über entsprechende Dienste sowohl in-game Informationen nutzen als auch externe Informationen innerhalb der virtuellen Welt zur Verfügung stellen können.

Im Folgenden wird zunächst beschrieben, wie die Anbindung externer Dienste an virtuelle Welten umgesetzt werden kann, danach werden Möglichkeiten der Integration von Dienste in die virtuelle Welt vorgestellt und bewertet. Im Anschluss wird die im Rahmen dieser Arbeit realisierte Erweiterung der Benutzerschnittstelle des Spiels zur Integration von Diensten in virtuelle Welten vorgestellt. Diese Erweiterung, das Service User Interface, ermöglicht eine angepasste Darstellung von Interaktionselemente und Servicedaten.

Anbindung externer Dienste

Die VCBS-Middleware stellt eine Verbindung her zwischen dem Client der virtuellen Welt und den externen Applikationen, die die Dienste bereitstellen. Der Informationsaustausch mit dem Client der virtuellen Welt anstelle des Servers der virtuellen Welt hat den Vorteil, dass beim Client sowohl alle benutzerbezogenen Daten (Kontextinformationen) vorliegen, als auch die Einbindung der Services benutzerspezifisch vorgenommen werden kann. Eine direkte Anbindung an den Spiele-Server ist aus Sicht der Spieleentwickler und -anbieter problematisch und wird deshalb nicht zugelassen. Eine solche Anbindung würde darüber hinaus ein zusätzliches Kommunikationsaufkommen zum Spiele-Server erzeugen. Eine zusätzliche Verbindung der VCBS-Middleware zum Server der virtuellen Welt ist optional vorgesehen, jedoch auf Zusatzinformationen, wie etwa zum Regelwerk für das Filtern von Kontextinformationen, beschränkt (siehe 5.4).

Abbildung 5.9: Darstellung von in-game Informationen außerhalb des Spiels am Beispiel der Charakterposition

In-game Informationen, die den Diensten im Spielumfeld bereitgestellt werden, können direkt durch Applikationen im Internet dargestellt werden. Ein Beispieldienst dafür ist eine Internet-Plattform, die bestimmte Charakterinformationen, wie die aktuelle Position des Benutzers in der virtuellen Welt, darstellt (siehe Abbildung 5.9).

Die Dienste können darüber hinaus die erhaltenen in-game Informationen weiterverarbeiten um Daten zu generieren, die der Benutzer in seiner aktuellen Situation benötigt. Die resultierenden Servicedaten (wie beispielsweise situationsabhängige Statistiken oder Guidelines) können über die VCBS-Middleware in die virtuelle Welt zurückübertragen und dort eingebettet dargestellt werden. Ein Beispieldienst, der beide Richtungen des Informationsflusses nutzt, ist der "*Fisherman's Friend*" Dienst (siehe auch 6.3.3). "Fisherman's Friend" unterstützt den Benutzer, wenn dieser in der virtuellen Welt angeln geht (siehe Abbildung 5.10). Der "Fisherman's Friend" Dienst wird dem Benutzer angeboten, wenn er in der Nähe eines virtuellen Gewässers ist und Angelutensilien (wie eine Angelrute) in der Hand hält. Sobald der Benutzer den Dienst verwendet, werden Informationen, wie etwa die Stelle, an der er angelt, und die Fische, die er fängt, über die VCBS-Middleware an den externen "Fisherman's Friend" Dienst übermittelt. Diese Informationen werden vom Dienst gesammelt und verwendet, um Statistiken über das Angeln zu berechnen. Zusätzlich werden die generierten Statistiken wie im ersten Fall (der Darstellung von Charakterinformationen über eine Internet-Plattform) auch im Internet dargestellt. Dann werden die Servicedaten wieder über die Middleware zurückübertragen und die Zusatzinformationen werden

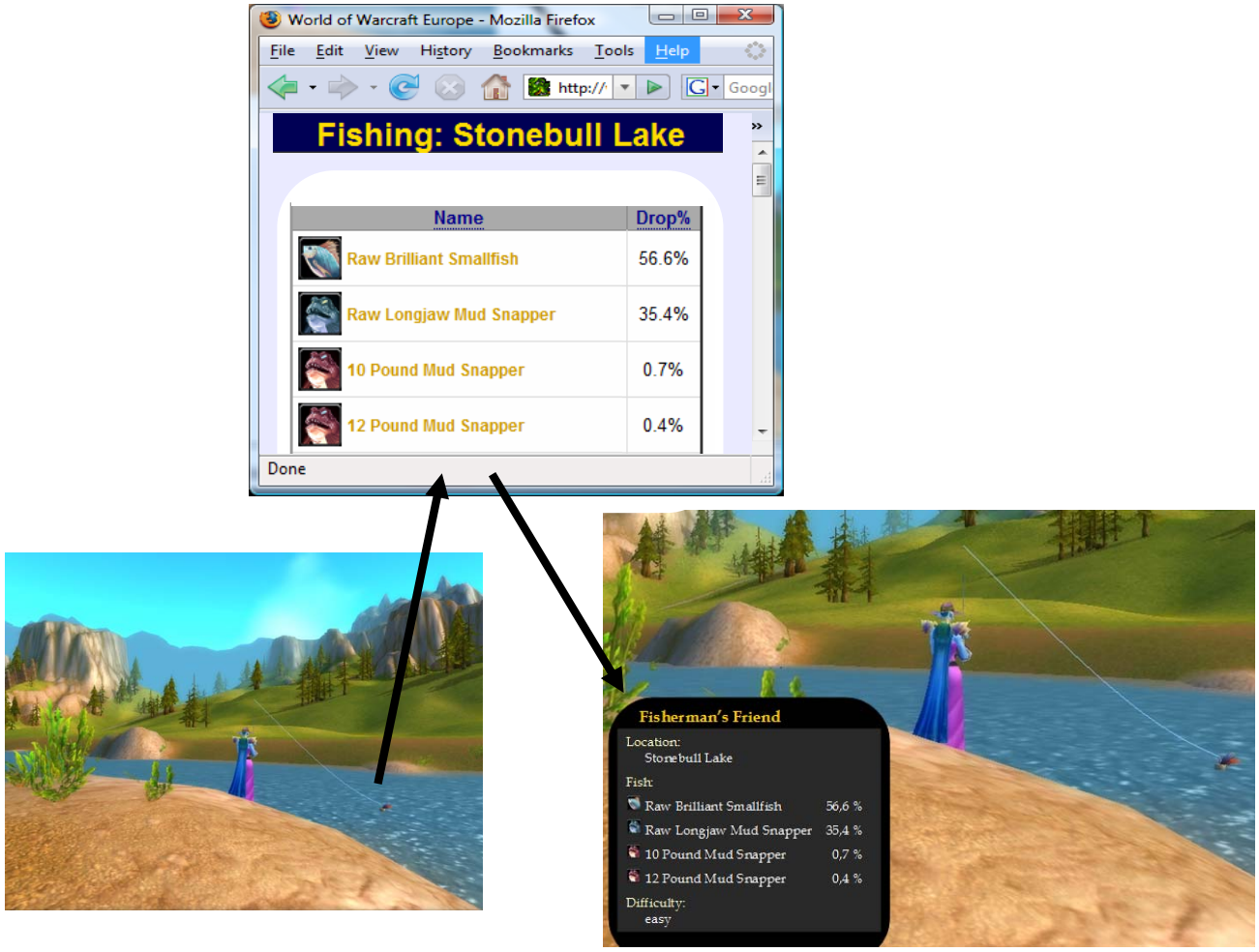

Abbildung 5.10: Nutzung von in-game Informationen zur Erzeugung von Servicedaten und Bereitstellung der Servicedaten innerhalb des Spiels am Beispiel "Fisherman's Friend"

dem Benutzer bereitgestellt. Dieser kann nun beispielsweise sehen, mit welcher Wahrscheinlichkeit verschiedene Fische gefangen werden können oder ob das Gewässer für seine Angelfähigkeiten geeignet ist.

Integration von Diensten in virtuelle Welten

Die Integration von Diensten in virtuelle Welten kann auf verschiedene Arten erfolgen. Eine Möglichkeit ist die Integration über das *Benutzerinterface* der virtuellen Welt. Dafür werden die externen Dienste als in-game Frames realisiert und, wie auch spielinterne Dienste (wie beispielsweise Minimap oder Questbuch), als Erweiterung des HUDs (Heads-Up Display) umgesetzt. Das HUD umfasst die spielinterne Anzeige aller Elemente, die dem Benutzer zusätzlich zur Darstellung der Spielwelt bereitgestellt werden. Das bedeutet, dass der Benutzer bei der Verwendung von Diensten neue Fenster oder *Icons* (Symbole) erhält, welche die Steuerelemente und Servicedaten des Dienstes repräsentieren (siehe auch Abbildung 5.10 als Beispiel dieser Art der Integration). Es ist dadurch möglich, verschiedene Typen von Servicedaten zu integrieren. Diese beinhalten textbasierte Daten, Bilder und Videodaten.

Eine weitere Möglichkeit ist die Integration in andere *Interface-Elemente wie Text- oder Voice-Chat*. Dabei können Text- oder Audiodaten mittels der vom Spiel zur Verfügung gestellten Kommunikationskanäle integriert werden. Bei Verwendung des Text-Chats können nur textbasierte Daten dargestellt werden und die Darstellung der Daten ist nur eingeschränkt möglich (zum Beispiel ohne große Formatierungsmöglichkeiten). Bei der Verwendung eines spielinternen Voice-Chats ist nur die Integration von Audiodaten

möglich. Eine Steuerung der Dienste kann mittels des in-game Chats realisiert werden. Dazu können im Fall des Text-Chats entsprechende Benutzereingaben verwendet werden, beispielsweise durch die Eingabe entsprechender Befehle in der Chat-Konsole (zum Beipiel "/activate Fisherman's Friend" um den "Fisherman's Friend" Dienst zu aktivieren). Im Fall des Voice-Chats wird für die Eingabe von Steuerbefehlen eine Spracherkennung auf Seiten der virtuellen Welt benötigt, die gesprochene Steuerbefehle erkennen kann. Voice-Chats sind jedoch aus Gründen der Spielperformanz nur selten überhaupt in virtuelle Welten integriert und eine entsprechende Spracherkennung ist von diesen Systemen nicht bekannt.

Eine andere Möglichkeit ist die Integration mittels einer *Visualisierung auf Spielobjekten*. Diese Art der Integration stellt einen größeren Eingriff in die Spielwelt dar, als die beiden vorgenannten Möglichkeiten. Sie erfordert das Vorhandensein von geeigneten Spielobjekten. Diese Spielobjekte müssen darüber hinaus adressierbar und für die Darstellung von Daten vorbereitet sein. Die Entwickler der virtuellen Welt müssen also diese Manipulationsmöglichkeiten an Spielobjekten vorsehen und auch die benötigten Manipulationen erlauben. Eine Darstellung von Servicedaten auf Spielobjekten kann beispielsweise durch die Projektion der Daten auf ein Spielobjekt oder durch den Einsatz von Texturen realisiert werden. Auch die Darstellung des Dienstangebots kann über eine Visualisierung auf Spielobjekten erfolgen. Zur Aktivierung und Deaktivierung der Dienste werden aber spezielle Bedienelemente benötigt, die geeignet mit dem Spielobjekt verknüpft werden müssen. Da die Darstellung der Spielwelt und ihrer Objekte durch den Spiel-Client durchgeführt wird, ist eine benutzerbezogene Darstellung möglich, auch wenn dabei das Aussehen von Spielobjekten verändert wird. Auch die Integration von Bedienelementen über Spielobjekte ist möglich, jedoch komplexer als die Integration der Servicedaten. Dafür müssen die Spielobjekte mit spielinternen Interaktionselementen versehen werden. Die Möglichkeiten dieser Interaktionselemente sind normalerweise auf ein Verwenden des entsprechenden Objektes beschränkt und bieten somit nur eingeschränkte Funktionalität. Die Eingabe von Annotationen durch den Benutzer ist beispielweise in diesem Fall nicht möglich. Eine Verwendung von Spielobjekten durch den Benutzer wird durch den Client der virtuellen Welt erfasst, benötigt in der Regel aber die Interaktion mit dem Spiel-Server, da dieser normalerweise die Aktionen der Benutzer auswertet. Somit würde eine Interaktion mit zusätzlichen Bedienelementen eine Kommunikation zwischen Spiel-Client und Spiel-Server erfordern, was ja eigentlich durch die Integration der Dienste in den Client der virtuellen Welt vermieden werden soll. Prinzipiell ist die Steuerung der Dienste über Spielobjekte möglich, deren Realisierung jedoch sehr aufwendig. Die dafür nötige Unterstützung durch die virtuelle Welt, inklusive der daraus resultierenden Manipulationsmöglichkeiten, auch in Anbetracht der potenzielle Sicherheitsrisiken, ist unrealistisch. Ein weiteres Problem dieser Art der Integration ist, dass Spielobjekte in den meisten Fällen örtlich gebunden sind. Das schränkt die Dienste, die integriert werden können, ein. Ein Beispiel ist die Anzeige von Zusatzinformationen zu einer *Quest*, die der Spieler gerade macht. Diese könnten auf einer spielinternen Informationstafel dargestellt werden. Der Spieler muss nun zum Nachlesen der Informationen immer wieder zu dieser Informationstafel zurückkehren und kann nicht jederzeit während der Erledigung der *Quest* auf die Zusatzinformationen zugreifen.

Auch eine Integration mittels der *Integration neuer Spielobjekte* kann umgesetzt werden. Für die Integration neuer Spielobjekte in die virtuellen Welt müssen entweder spezielle Spielobjekte implementiert sein, die für einen Dienst instanziiert werden können, oder es muss die Möglichkeit geben, neue Inhalte (*user created content*) in die virtuelle Welt zu integrieren. Es handelt sich in diesem Fall um einen noch tieferen und komplexeren Eingriff in die virtuelle Welt, als bei der Visualisierung auf bereits bestehenden Spielobjekten. Die entstehenden Problematiken sind jedoch dieselben. Der Vorteil der Integration neuer Spielobjekte ist, dass diese nun beliebig an jeder Stelle der virtuellen Welt integriert werden können und so beispielsweise die Informationstafel aus dem oben genannten Beispiel zu jeder Zeit neben dem Spieler erscheinen kann. Dies führt jedoch dazu, dass nun zusätzliche Steuerelemente benötigt werden, um die Darstellung der Dienstdaten abzurufen, da sich ansonsten die Informationstafel mit dem virtuellen Charakter zusammen durch die virtuelle Welt bewegen müsste, solange dieser Dienst aktiv und gültig ist. Auch für das Angebot verfügbarer Dienste besteht die gleiche Problematik wie bei der Visualisierung auf bestehenden Spielobjekten.

Die Integration über das Benutzerinterface der virtuellen Welt ermöglicht die Darstellung der Servicedaten innerhalb der Darstellung des Clients der virtuellen Welt und somit die parallele Visualisierung von Spielwelt und Servicedaten. Darüber hinaus wird die Interaktion mit den externen Diensten innerhalb des Spiel-Clients (ohne Taskwechsel) ermöglicht. Die Darstellung der Dienste kann im Stil der Spieldarstellung erfolgen und für den Benutzer ohne Unterschied zu spielinternen Diensten umgesetzt werden. Vorteile der Integration über das Benutzerinterface, sind die dadurch ermöglichte Kontrolle der Dienstverwendung durch den Benutzer und die Flexibilität bei der Darstellung der Daten. Da die von der Middleware bereitgestellten kontextbasierten Dienste dem Benutzer angeboten werden, sobald sie für ihn verfügbar sind und der Benutzer dann entscheidet, ob er die angebotenen Dienste auch verwenden möchte, ist die Realisierung geeigneter Bedienelemente zur Steuerung der Dienste durch den Benutzer nötig. Mittels entsprechender Bedienelemente werden sowohl die Aktivierung und Deaktivierung verfügbarer Dienste, als auch die Interaktion des Benutzers mit den externen Diensten ermöglicht. So können beispielsweise auch Benutzereingaben erfasst werden und Dienste wie der "lokationsbasierte Annotationsdienst" (siehe auch 6.3.4) integriert werden, der die Dokumentation der virtuellen Umgebung durch Annotationen unterstützt, die der Spieler in der virtuellen Umgebung hinterläßt. In dieser Arbeit steht die Integration von textbasierten Servicedaten im Vordergrund. Auch aus diesem Grund ist die Darstellung als in-game Frames über das Benutzerinterface die geeignetste Alternative, da diese eine Formatierung der Daten ermöglicht.

Auch eine Kombination verschiedener Integrationsformen ist denkbar. Um die manuelle Aktivierung von Diensten durch den Benutzer umzusetzen, ist als Basis einer solchen Kombination in jedem Fall eine Integration über das Benutzerinterface sinnvoll, da über das Benutzerinterface verschiedene Bedienelemente zur Steuerung der Dienste realisiert werden können. Eine Bedienung der Dienste über den Text-Chat ist auch möglich, diese ist jedoch für den Benutzer umständlich, da er dabei mit Konsolenbefehlen arbeiten muss und auch keine Interaktionselemente wie *Buttons* oder *Drop-down* Menüs eingesetzt werden können. Die Integration der Servicedaten kann auch getrennt von den Bedienelementen erfolgen. Auch wenn die Steuerung der Dienste über das Benutzerinterfache erfolgt, können textbasierte Servicedaten beispielsweise durch die Visualisierung auf Spielobjekten oder unter Verwendung des ins Spiel integrierten Text-Chats dargestellt werden. Vor allem für eine Integration von Video- oder Audiodaten kann die getrennte Darstellung sinnvoll sein, beispielsweise durch die Projektion von Videodaten auf bestimmte Spielobjekte oder die Integration von Audiodaten in einen spielinternen Voice-Chat. Auch Servicedaten, die nicht direkt dargestellt werden, sondern beispielsweise der Steuerung von Diensten dienen, wie bei einem lokationsbasierten Voice-Chat (siehe 6.3.1 und [SA04]), werden nicht über das Benutzerinterface in die virtuelle Welt integriert.

Integrierte Darstellung durch das Service User Interface

Die Einbindung der externen Dienste in die virtuelle Umgebung über die VCBS-Middleware geschieht durch eine Erweiterung des Benutzerinterfaces des Spiel-Clients. Für die Kontrolle der Dienstverwendung durch den Benutzer wurde in dieser Arbeit das Konzept des Service User Interfaces (SUI) entwickelt. Das Service User Interface wird als Erweiterung des Benutzerinterfaces der virtuellen Welt realisiert, so dass eine Interaktion mit dem Service User Interface während des Aufenthalts in der virtuellen Umgebung ermöglicht wird. Über das Service User Interface können verschiedene Funktionalitäten innerhalb der virtuellen Welt bereitgestellt werden. Dabei sind die wichtigsten Funktionen die Repräsentation der aktiven Dienste, das Angebot der zur Verfügung stehenden Dienste und die Aktivierung und Deaktivierung von Diensten durch den Benutzer.

Das Service User Interface bildet somit die eigentliche Schnittstelle zwischen dem Benutzer und der VCBS-Middleware. Eine konkrete Instanz ist jedoch abhängig von der virtuellen Welt, da deren Darstellungsart eingehalten werden muss, um eine nahtlose Integration zu erreichen. Abbildung 5.11 zeigt eine mögliche Umsetzung eines Service User Interfaces. Neben den Basisfunktionalitäten können optio-

nal weitere Features umgesetzt werden. Das dargestellte Service User Interface beinhaltet eine Kategorie "*In Use*", die eine Übersicht aller Dienste gibt, die der Benutzer gerade verwendet, und den Zugriff auf zusätzliche Informationen, zum Beispiel welche Kontextinformation verwendet werden, über diese Dienste ermöglicht. "*Available*" beinhaltet alle Dienste, die dem Benutzer angeboten werden, die er jedoch nicht aktiviert hat. In den "*Favorites*" kann der Benutzer Dienste festlegen, die er immer - wenn möglich - benutzen möchte. Diese werden dann automatisch aktiviert, sobald sie verfügbar sind. Unter "*Configuration*" finden sich Optionen zur Konfiguration des Service User Interfaces, wie die Anpassung der Darstellungsweise. Ein weiteres optionales Feature ist die *Ignorelist*, auf der vom Benutzer Dienste eingetragen werden, die er niemals nutzen möchten und die ihm dann nicht mehr angeboten werden.

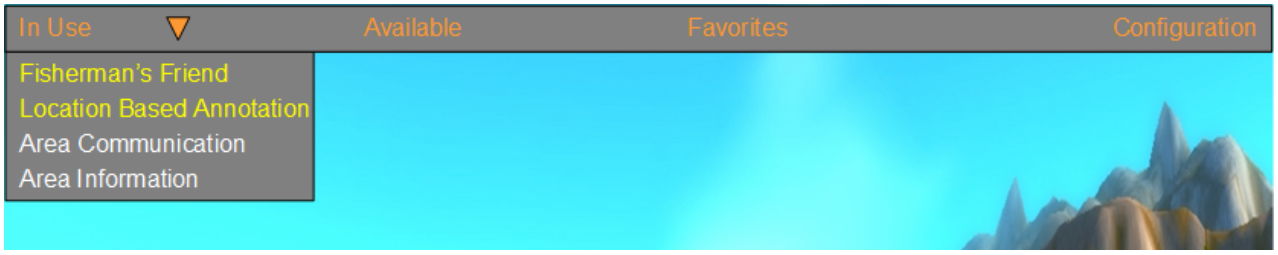

Abbildung 5.11: Beispiel eines Service User Interfaces (SUI)

Der große Vorteil des Service User Interfaces für den Benutzer ist, dass ihm darüber jederzeit die für ihn aktuell in Frage kommenden Dienste, direkt im Spiel angeboten werden und er dort diese Dienste aktivieren oder deaktivieren kann. Er ist somit immer darüber informiert, was es für Dienste für seine aktuelle Situation gibt und kann dann selbst entscheiden, welche er nutzen möchte.

5.4 Realisierung non-funktionaler Anforderungen durch die VCBS-Middleware

Auch die in Abschnitt 5.1.2 dargestellten non-funktionalen Anforderungen werden in der VCBS-Middleware berücksichtigt. Es sind die *Datensicherheit* (Cheating und Privacy), sowie *Benutzerfreundlichkeit* inklusive Beteiligung der Community. Im Folgenden werden diese Aspekte genauer dargestellt. In puncto Datensicherheit sind Methoden zum Schutz der Privatsphäre des Benutzers (Privacy) sowie zum Schutz der Spiele (Cheating) in der VCBS-Middleware realisiert. Um die Bedürfnisse der Benutzer bestmöglich unterstützen zu können, ermöglicht es die VCBS-Middleware der Community, eigene Dienste zu entwickeln, die dann über die VCBS-Middleware genutzt werden können.

5.4.1 Datensicherheit - Cheating und Privacy

Im Bereich Datensicherheit gibt es vor allem zwei wichtige Anforderungen (siehe Abschnitt 5.1.2), die auch in dieser Arbeit berücksichtigt werden. Sicherheit ist ein wichtiges Thema im Bereich der MOGs wie auch bei der Verwendung von Kontextinformationen, da es sich hierbei um personenbezogene Daten handelt. Zum einen muss die Middleware für den Anbieter des Spieles die Sicherheit bieten, dass nur dann Kontextinformationen weitergegeben werden, wenn dies auch im Sinne des Spiels ist, damit eine missbräuchliche Verwendung "*Cheating*" verhindert werden kann. Zum anderen muss für den Spieler, der die VCBS-Middleware nutzt, die Sicherheit bestehen, dass seine personenbezogenen Daten nicht in einer Art verwendet werden, die er nicht möchte ("*Privacy*"). Virtuelle Kontextinformationen werden zwar bezüglich des virtuellen Charakters erhoben, da dieser aber eine Repräsentation des realen Benutzer darstellt, handelt es sich hier um eine spezielle Form von personenbezogenen Daten.

Cheating

Ein nicht begrenzter Informationsaustausch zwischen einem Spiel und anderen Internet-Applikationen eröffnet nicht nur viele neue Möglichkeiten, ein unbeschränkter Zugriff auf in-game Informationen ermöglicht auch deren Missbrauch (siehe auch Abschnitt 5.1.2). Um negative Auswirkungen und Angriffe so wie unfaire Vorteile für einzelne Spieler oder die Verwendung von unerlaubten Hilfsmitteln zu verhindern, muss ein Schutzmechanismus implementiert werden, der es ermöglicht, den Informationsaustausch zu steuern.

Um einer missbräuchlichen Benutzung dieser Informationen entgegenzuwirken, werden von der VCBS-Middleware nicht alle Kontextinformation zu jeder Zeit verfügbar gemacht. Beispielsweise ist die Anzeige der Position eines Spielers außerhalb des Spiels für die Verknüpfung von Mitspielern hilfreich. Begibt sich der Spieler jedoch in einen Wettkampf, so wird die Anzeige seiner Position zum Nachteil, da die Gegenspieler nun diese Information zu ihrem Vorteil nutzen können. Um dies zu unterbinden und um trotzdem in der Lage zu sein, nützliche Informationen weiterzugeben, filtert die VCBS-Middleware die an die Dienste weitergeleiteten Kontextinformationen. Dieser Filtermechanismus verwendet die virtuellen Kontextinformationen, da sie die virtuelle Situation des Benutzers beschreiben. Es wird also die virtuelle Kontextinformation, basierend auf virtuellem Kontext gefiltert. Dadurch kann entschieden werden, welche Kontextinformationen über die virtuelle Situation an das Umfeld weitergegeben werden können und welche nicht.

Die Grundlage für das situationsabhängige Filtern der Kontextinformationen bilden Regelwerke. Virtuelle Welten basieren auf Regelwerken, die die virtuellen Umgebungen definieren. Durch virtuelle Parameter können die Regelwerke in der gleichen Granularität beschrieben werden, wie die Kontextinformationen. Beispielsweise kann eine solche Regel definieren, dass die virtuelle Position des virtuellen Charakters eines Spieles nur weitergegeben wird, wenn sich der Charakter nicht im Kampf mit anderen Spielern befindet. Daher wird abhängig von der Situation des virtuellen Charakters bestimmt, ob seine in-game Position an das Spielumfeld weitergegeben wird oder nicht. Nur wenn der Charakter sich nicht im Kampf befindet (er ist zum Beispiel im Peace-Mode) oder gegen Computergegener kämpft (PVE), dürfen seine Positionsinformationen weitergegeben werden.

Die Regelwerke, die für das Filtern verwendet werden, können von den Spieleherstellern (oder Betreibern) definiert werden. Dadurch haben die Spielebetreiber die Möglichkeit, den Informationsfluss aus dem Spiel heraus zu steuern, was einen großen Anreiz bietet, überhaupt Informationen von außen zugreifbar zu machen. Es ist sogar notwendig, den Spielbetreibern diese Möglichkeit zu bieten, da sie letztendlich dafür Verantwortung tragen, dass ein Spiel fair bleibt und alle Spieler die gleichen Chancen haben (*balancing*). Würde man einfach alle Informationen zu jedem Zeitpunkt zugreifbar machen, dann könnte man keine Fairness mehr garantieren und würde dem Spiel schaden und im schlimmsten Fall das Gameplay und somit das Spiel ruinieren.

Privacy

Die kontextbasierten Dienste werden dem Benutzer über ein Service User Interface angeboten, falls er sich in einem entsprechenden virtuellen Kontext befindet. Der Benutzer kann dann über die Verwendung des Dienstes entscheiden. Es werden ihm Informationen über die angebotenen Dienste bereitgestellt, beispielsweise welche Kontextinformationen an die Dienste weitergegeben werden. Der Benutzer kann daraufhin entscheiden, ob er diesen Dienst in Anspruch nehmen möchte oder nicht. Das ist ein wichtiger Aspekt der VCBS-Middleware. Die Benutzer entscheiden über die Verwendung der Dienste und gleichzeitig darüber, welche seiner virtuellen Kontextinformationen zur Verfügung gestellt werden. Wenn ein Benutzer beispielsweise nicht möchte, dass Informationen über seinen aktuellen Aufenthaltsort im Spiel an andere Benutzer weitergegeben werden, dann wird er einen entsprechenden Dienst nicht benutzen.

Der Benutzer muss in der Lage bleiben, zu entscheiden, welche seiner personenbezogenen Kontextinformationen weitergegeben werden. Aus diesem Grund ist es nötig, zum einen den Benutzer entsprechend zu informieren und zum anderen den Benutzer entscheiden zu lassen ("*leave the human in the control loop*" [Eri02]), auch wenn dadurch zusätzliche Benutzeraktionen nötig werden und eine völli-

ge Automation bequemer wäre. Ein wichtiger Grund für dieses Vorgehen ist, dass der Benutzer immer die Kontrolle über seine Kontextinformationen haben sollte [ST93] und deshalb darüber entscheiden können muss, welcher Dienst Zugriff auf seine Kontextinformationen erhält [CK00].

Um die Privatsphäre des Benutzers zu schützen, werden die Dienste außerdem nur lose an die VCBS-Middleware gekoppelt. Die Dienste kennen dadurch nur die virtuelle Identität des Benutzers und können keine Rückschlüsse auf den realen Benutzer ziehen. Benutzerverwaltung und Authentizifizierung der Benutzer werden von der VCBS-Middleware durchgeführt.

5.4.2 Benutzerfreundlichkeit und Beteiligung der Community

Ein weiteres Ziel der VCBS-Middleware und non-funktionale Anforderung ist es, eine aktive Beteiligung der Community zu ermöglichen. Die Beteiligung der Community ist ein wichtiger Faktor und sollte bei allen neuen Entwicklungen für MOGs nicht mehr vernachlässigt werden. Die aktive Beteiligung der Community ist wichtig für die Weiterentwicklung vernetzter Spiele (siehe auch Abschnitt 3.3.3). Aus diesem Grund eröffnet die VCBS-Middleware jedem die Möglichkeit, selbst neue Dienste zu entwickeln. Die von der Community entwickelten Dienste können dann über die VCBS-Middleware mit dem Spiel verknüpft und in die virtuelle Umgebung eingebettet werden.

Dadurch ergeben sich neue Formen der Community-Beteiligung. Die Community kennt ihren eigenen Bedarf am besten. So entstehen Dienste, die wiederum für die Community nützliche Unterstützungen im Spiel anbieten. Auch variablen Anforderungen bezüglich einer Unterstützung im Spiel durch verschiedene Typen von Spielern kann so begegnet werden. Spieler können ganz unterschiedliche Spielgewohnheiten haben (wie Spieldauer, Spielhäufigkeit oder Spielintensität) und spielen mit unterschiedlichen Intentionen (wie das Erreichen von spielbezogenen Zielen, Erkundung der Spielwelt, Sozialisierung oder Unruhe stiften [Bar96]). Verschiedene Spielertypen (wie *Power Gamer* oder *Casual Gamer* [Tay03]) verfolgen dabei in der Regel unterschiedliche Strategien und Spielziele. Diese verschiedenen Spielertypen benötigen daher teilweise unterschiedliche Formen der Unterstützung. *Power Gamer* spielen in der Regel sehr effizient und führen detaillierte Analysen von Spielsituationen durch. Ein Dienst, der Spielsituationen protokolliert, sodass diese nachträglich aufgearbeitet werden können, ist vor allem für *Power Gamer* relevant. *Casual Gamer* hingegen probieren viele Dinge einfach erstmal aus und sind oft nicht bereit, sich in alle Details der Spielmechanik einzuarbeiten. Ein Dienst, der Hilfestellungen anbietet, ohne auf die Details einzugehen, beispielsweise einfach nur anzeigt, welcher Ausrüstungsgegenstand sich für einen Charakter besser eignet, ist vor allem für *Casual Gamer* interessant. Eine Unterstützung innerhalb des Spiels ist jedoch für alle Spielertypen relevant, nur die benötigten Dienste können variieren. Dabei wird eine Vielfältigkeit an Diensten durch eine Beteiligung der Community und die Nutzung des dort vorhandenen kreativen Potentials erreicht.

Ein weiterer Vorteil der flexiblen Diensterstellung, den die VCBS-Middleware bietet, besteht darin, dass die Schnittstelle für die externen Dienste einheitlich verwendet werden kann, da sie für alle Spiele dieselben Grundlagen bietet. Egal für welches MOG ein Dienst entwickelt wird, das verwendete Prinzip und die eingesetzten Mechanismen sind immer dieselben. Dadurch müssen nicht für jedes Spiel neue Tools, APIs oder Programmiersprachen erlernt werden.

6 Implementierung

In diesem Kapitel wird die Implementierung der VCBS-Middleware sowie der weiteren Komponenten beschrieben, die zum Nachweis der Umsetzbarkeit des Konzepts der virtuell kontextbasierten Dienste entwickelt wurden. Dabei wird vor allem auf die Funktionsweise und den Informationsfluss, der durch die Middleware realisiert wird, eingegangen. Danach wird die Anbindung der Spiele vorgestellt, die für die Validierung der Middleware eingesetzt werden, und im Anschluß werden die entwickelten Beispieldienste beschrieben.

6.1 VCBS-Middleware

Die Funktionsweise der im letzten Kapitel vorgestellten Komponenten (Client-Komponente und Online-Komponente, siehe 5.3.2) werden nun im Detail betrachtet. Dabei ist der wichtigste Aspekt der Informationsfluss zwischen Spiel-Client und externen Applikationen, den die VCBS-Middleware ermöglicht (siehe Abbildung 6.1). Die virtuellen Kontextinformationen werden von der *Client-Komponente* der Middleware erfasst (1) und an die *Online-Komponente* übertragen (2).

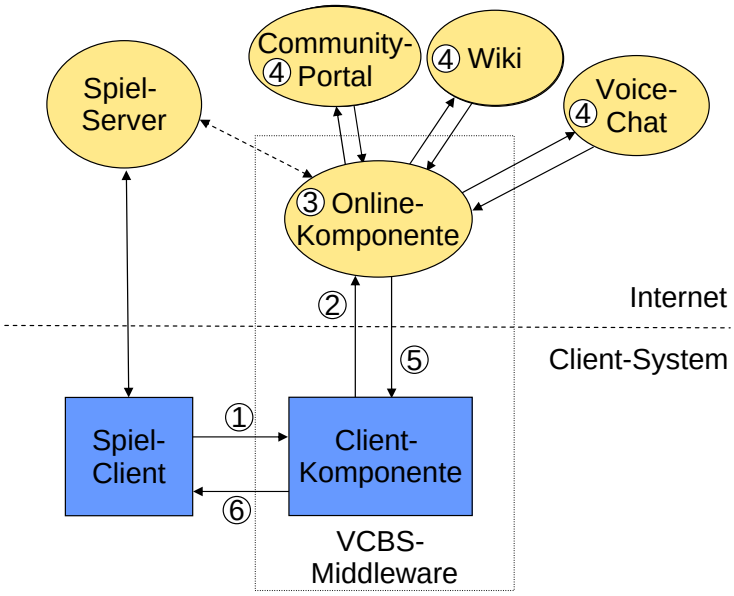

Abbildung 6.1: Informationsfluss zwischen Spiel-Client und externen Internet-Applikationen über die VCBS-Middleware

In der Online-Komponente werden die Kontextinformationen ausgewertet und die zu der Situation passenden Dienste ermittelt (3). Die Kontextinformationen des Benutzers werden entsprechend der Spezifikation der verwendeten Dienste gefiltert und die jeweils benötigten Informationen an die externen Dienste weitergeleitet, wo sie verarbeitet und / oder dargestellt werden (4). Die Daten der externen Dienste (Servicedaten), die zur Realisierung der Dienste in der virtuellen Welt benötigt werden, werden an die Online-Komponente der VCBS-Middleware zurückübermittelt, welche diese Daten dann an die Client-Komponente weiterüberträgt (5). Die Client-Komponente bettet die erhaltenen Servicedaten dann in den Client der virtuellen Welt (*Spiel Client*) ein (6).

6.1.1 Komponenten der VCBS-Middleware und Darstellung ihrer Funktionen

Die Client- und die Online-Komponente setzten die in Kapitel 5.3.1 definierten Funktionen um. Zur Realisierung der verschiedenen Funktionen sind diese in weitere Komponenten unterteilt, die eigenständige Funktionsblöcke darstellen und auf verschiedene Rechnersysteme verteilt werden können. Abbildung 6.2 gibt einen Überblick über die weiteren einzelnen Bestandteile der VCBS-Middleware.

Die Client-Komponente besteht aus dem **VCBS-Client**. Dieser befindet sich auf dem System des Benutzers und ist das Bindeglied zwischen dem Client einer virtuellen Welt und der Online-Komponente der VCBS-Middleware. Der VCBS-Client erfüllt drei Funktionen, die *Benutzter Authentisierung*, die *Kontextaquise* und die *Dienstintegration*. Zur Anbindung bestehender *Spiele-Clients* kann der VCBS-Client durch **Spiel Plug-Ins** erweitert werden.

Die Online-Komponente erfüllt Aufgaben der zwei Funktionsbereiche Informationsverarbeitung und Verwaltung. Der Bereich Informationsverarbeitung umfasst die *Kontextverarbeitung* und das Filtern von Kontextinformationen für die *Kontextverteilung*, sowie das auf dem Kontext basierende *Dienst-Matching*, und die benutzerbasierte *Dienstaktivierung*. Die Funktionen der Informationsverarbeitung werden vom **VCBS-Server** durchgeführt. Die Verwaltungsfunktionen werden von der zentralen *Management*-Instanz der VCBS-Middleware, der **VCBS-Registry** realisiert. Sie umfassen die *Dienstverwaltung*, die *Benutzerverwaltung* und die Verwaltung der verschiedenen VCBS-Server. Für die Anbindung der *externen Dienste*, von verschiedenen Internet-Applikationen, wird das **VCBS-Interface** bereitgestellt.

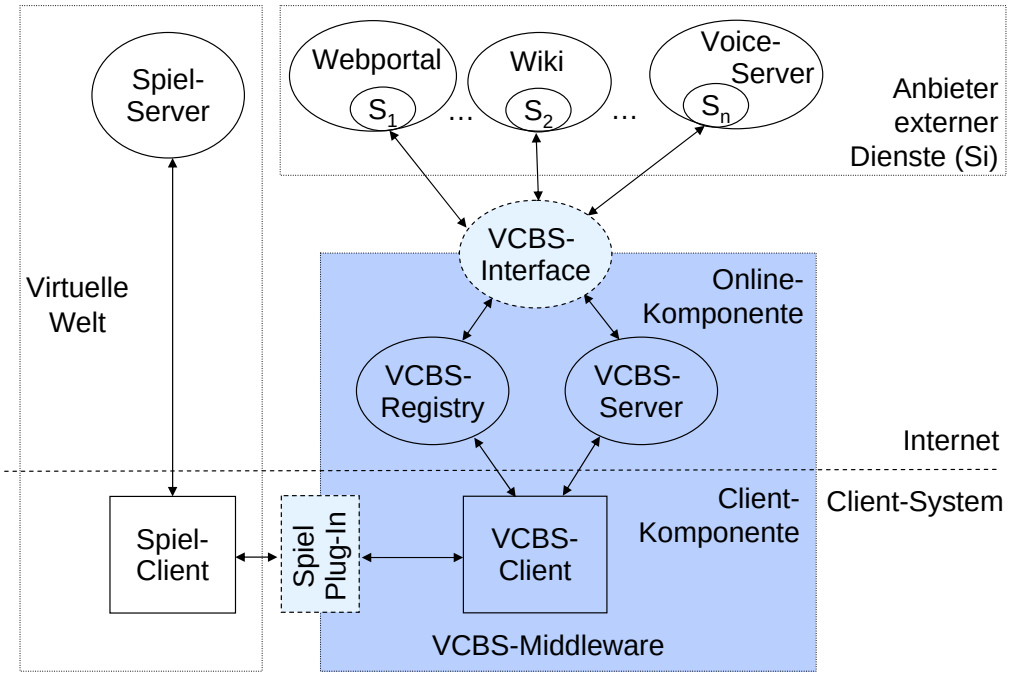

Abbildung 6.2: Überblick über die Komponenten der VCBS-Middleware

Im Folgenden werden die Bestandteile der VCBS-Middleware genauer betrachtet. Besonders die Bestandteile, die für die Verarbeitung und Weitergabe der Informationen eingesetzt werden, werden detailliert vorgestellt. Abbildung 6.3 zeigt eine Übersicht des Aufbaus der einzelnen Komponenten der VCBS-Middleware. Die dicken Pfeile stellen den Fluss der Informationen durch die VCBS-Middleware dar.

VCBS-Client

Der VCBS-Client ist auf dem System des Benutzers installiert und unterhält eine Verbindung zu dem aktiven Spiel. Für die Verbindung mit einem Spiel werden *Spiel Plug-Ins* verwendet, um Informationen

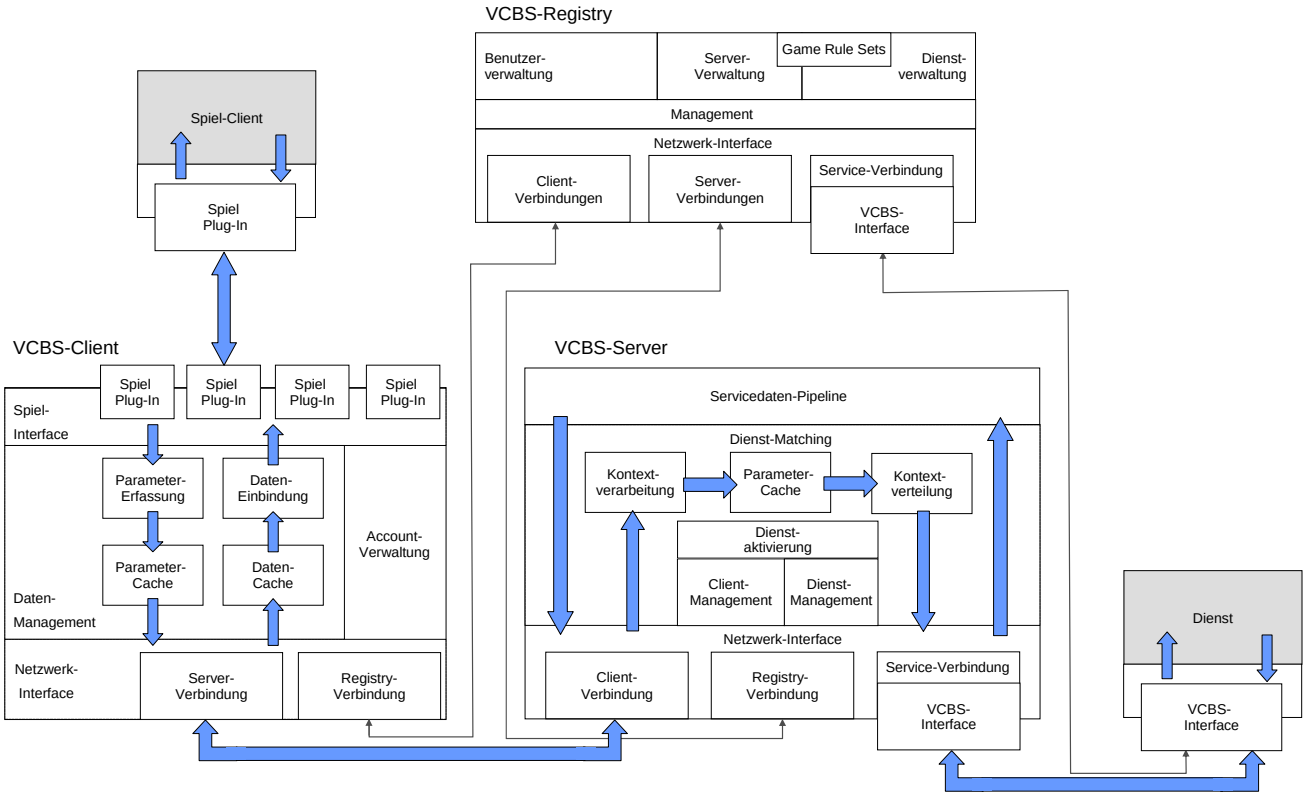

Abbildung 6.3: Architektur-Überblick der Komponenten der VCBS-Middleware

mit dem Spiel-Client auszutauschen. Verschiedene spielspezifische Plug-Ins ermöglichen die Verbindung mit verschiedenen Spiele-APIs, die von den einzelnen Spieleanbietern bereitgestellt werden. Die Plug-Ins ermöglichen über die APIs einen Austausch von virtuellen Kontextinformationen und externen Servicedaten. Aus diesem Grund ist der VCBS-Client modular aufgebaut und einfach zu erweitern. Die Basiskomponenten, die der VCBS-Client bereitstellt, ermöglichen es, die Komplexität und den Aufwand bei der Erstellung neuer Plug-Ins gering zu halten. Die *Account-Verwaltung*, das *Daten-Management* und das *Netzwerk-Interface*, stellen die benötigten Funktionalitäten bereit, die durch die spielspezifischen Plug-Ins ergänzt werden. Zusätzlich enthält der VCBS-Client eine graphische Benutzerschnittstelle, welche die installierten Plug-Ins anzeigt und zur Kontrolle durch den Benutzer einen Zugriff auf deren Daten bietet.

Die Account-Verwaltung speichert die benutzerspezifischen Angaben und Einstellungen. Wichtige Funktionalitäten des Daten-Managements sind die Informationserfassung und -konvertierung (*Parameter-Erfassung*) sowie das Caching der Informationen (*Parameter-Cache*). Nach der Erfassung der in-game Informationen über ein Spiel Plug-In des VCBS-Clients werden die Kontextinformationen in die xgdl-Datenstruktur konvertiert und gespeichert. Die erhaltenen Kontextinformationen müssen, soweit sie noch nicht im xgdl-Format vorliegen, in dieses konvertiert werden. Aufgabe des Netzwerk-Interfaces ist die Durchführung der Kommunikation mit den anderen Teilen der VCBS-Middleware. Der VCBS-Client nimmt die vom Spiel-Client erfassten Kontextinformationen und überträgt diese an den zugewiesenen VCBS-Server. Treten Änderungen der virtuellen Parameter auf, dann werden die geänderten Parameter über das Netzwerk-Interface des VCBS-Clients an den VCBS-Server übertragen.

Der VCBS-Server informiert den VCBS-Client über verfügbare gültige Dienste. Diese werden über das Spiel-Interface in die virtuelle Welt integriert und dem Benutzer angeboten. Der Benutzer kann gültige Dienste aktivieren und verwenden. Für die aktiven Dienste erhält der VCBS-Client die vom Dienst bereitgestellten Daten (Servicedaten) über den VCBS-Server. Die empfangenen Servicedaten werden im *Daten-Cache* gespeichert und über das *Spiel Plug-In* in den Client der virtuellen Welt eingebunden (*Daten-Einbindung*).

Technisch findet die Integration der Daten (Dienstangebote und Servicedaten) in das Spiel über das Benutzerinterface des Spiel-Clients statt (siehe auch Kapitel 5.3.3). Für die Umsetzung der Integration von Diensten in virtuelle Umgebungen wird ein Service User Interface verwendet (siehe auch 5.3.3).

VCBS-Server

Der VCBS-Server verarbeitet die Kontextinformationen (*Kontextverarbeitung*). Er verwendet die Kontextinformationen, die er von den VCBS-Clients erhält, um Dienste zu ermitteln, die in der virtuellen Situation eines Benutzers gültig sind (*Dienst-Matching*). Alle aktuell registrierten Dienste bekommt der VCBS-Server von der VCBS-Registry mitgeteilt und kennt somit deren Beschreibungen. Die Informationen über die verfügbaren Dienste werden im(*Dienst-Management*) verwaltet. Gültige Dienste werden dem Benutzer angeboten. Diejenigen Dienste, die für einen Benutzer gültig sind und diejenigen Dienste, die der Benutzer aktiviert hat, verwaltet das *Client-Management*. Die *Dienstaktivierung* steuert die Aktivierung und Deaktivierung der Dienste durch den Benutzer oder falls diese ungültig werden. Die Kontextparameter werden nach der Verarbeitung im *Parameter-Cache* gespeichert und wenn aktive Dienste vorliegen, werden die relevanten Kontextparameter an die externen Dienste weitergeleitet (*Kontextverteilung*). Die Auswahl der Informationen, die an den Dienst weitergegeben werden, sind abhängig vom jeweiligen Dienst. Für den "Fisherman's Friend" Dienst beispielsweise umfassen die benötigten Informationen die Lokation und die Ausrüstung des virtuellen Charakters.

Über das *VCBS-Interface* empfängt der Dienst die für ihn erforderlichen Kontextinformationen und hat die Möglichkeit, von ihm generierten Servicedaten zurück an den VCBS-Server zu übertragen. Der Dienst verwendet die erhaltenen Informationen beispielsweise, indem er daraus Angel-Statistiken generiert und diese über eine Web-Applikation im Internet zugänglich macht. Zusätzlich sendet er die ermittelten Informationen als Servicedaten über das VCBS-Interface an den VCBS-Server zurück. Die Servicedaten werden dann über die *Servicedaten-Pipeline* an den VCBS-Client weitergeleitet, vom VCBS-Client in das Spiel eingebettet und innerhalb der virtuellen Welt dargestellt. Beim "Fisherman's Friend" Dienst umfassen die Servicedaten, die dem Benutzer in der virtuellen Welt bereitgestellt werden, beispielsweise die Bezeichnung des aktuellen Angelplatzes, die Wahrscheinlichkeiten der verschiedenen Fische, die an dieser Stelle gefangen werden können, sowie die Anforderungen an die Angelfähigkeit (Schwierigkeitsgrad des Gewässers, siehe auch Abbildung 5.10). Der VCBS-Server überprüft regelmäßig, basierend auf den Kontextinformation des Clients, ob die Dienste, die der Benutzer gerade verwendet, noch gültig sind und ob neue Dienste verfügbar sind. Ist dies der Fall, werden diese neu verfügbaren Dienste dem Benutzer angeboten. Hat ein Benutzer beispielsweise seine Position verändert und befindet sich nun neben einem virtuellen See und ist er zusätzlich mit einer Angel ausgerüstet, dann erfüllt er die Voraussetzungen für den "Fisherman's Friend" Dienst. Dieser jetzt neu gültige Dienst wird dem Benutzer angeboten. Der Benutzer entscheidet dann, ob er den angebotenen Dienst aktivieren möchte oder nicht. Verwendet der Benutzer nun den "Fisherman's Friend" Dienst, ändert dann aber später seine Ausrüstung indem er ein Schwert anstelle der Angel in die Hand nimmt, teilt der Client dies dem VCBS-Server anhand der Kontextinformationen mit. Wenn ein Dienst ungültig wird, weil sich die virtuelle Situation des Benutzers ändert oder wenn der Benutzer einen Dienst nicht mehr verwenden möchte und ihn deaktiviert, dann beendet auch der Server die Weitergabe der Kontextinformationen an den externen Dienst. Der VCBS-Client wird dann vom Server darüber informiert, dass der Dienst nun nicht mehr gültig ist und der Dienst wird beendet.

VCBS-Registry

Die VCBS-Registry führt alle administrativen Aufgaben der VCBS-Middleware durch. Sie verwaltet die Benutzer der Middleware (*Benutzerverwaltung*), die externen Dienste, die über die Middleware angeboten werden (*Dienstverwaltung*) und die verschiedenen VCBS-Server (*Server-Verwaltung*). In der VCBS-Middleware werden mehrere VCBS-Server verwendet, um die entstehende Last auf verschiedene Systeme zu verteilen. Die Registry übernimmt neben An- und Abmeldung der Benutzer auch deren Zuweisung

zu den verschiedenen VCBS-Servern (ein Benutzer bekommt für seine Session einen festen VCBS-Server zugewiesen).

Um die VCBS-Middleware nutzen zu können, benötigt der Benutzer einen VCBS-Account. Über den VCBS-Client kann sich ein Benutzer mit seinem Account an- und abmelden. Um die Privatsphäre des Benutzer zu schützen, werden nur für angemeldete Benutzer Informationen gesammelt und übertragen.

Um einen Dienst über die VCBS-Middleware anbieten zu lassen, muss dieser zuerst registriert werden. Dafür stellt die Registry eine *Service-Verbindung* bereit. Die Beschreibung eines registrierten Dienstes wird an die VCBS-Server verteilt und im Anschluss kann der Dienst den Benutzern angeboten werden. Bei der Registrierung der Dienste werden sie durch die Registry anhand hinterlegter Regelwerke (*Game Rule Sets*) auf ihre Gültigkeit überprüft. Informationen über neue Dienste oder Updates werden an die Server weitergeleitet. Die Registry verwendet Status-Updates, um die Server immer auf dem aktuellen Stand zu halten.

6.1.2 Dienstbeschreibung und Dienst-Matching

Im Folgenden wird die Beschreibung von Diensten für die VCBS-Middleware vorgestellt und die Durchführung des Dienst-Matching an einem Beispiel erläutert.

Die Frage nach einer geeigneten **Dienstbeschreibung** stellt sich im Rahmen dieser Arbeit für Dienste, die auf virtuellem Kontext basieren, da noch keine entsprechenden Beschreibungen verfügbar sind. Daraus ergibt sich jedoch auch die Chance, die verwendeten Dienstbeschreibungen auf die Eigenschaften von virtuellem Kontext anzupassen. Dienste, die auf virtuellem Kontext basieren, können direkt auf Basis einer xgdl-basierten Kontextbeschreibung beschrieben werden. Die in dieser Arbeit entwickelte eXtensible Game Description Language (xgdl) definiert eine Beschreibung von virtuellem Kontext, die direkt auf virtuellen Parametern basiert und in-game Kontextinformationen in einem allgemeinen Format austauschbar macht (siehe Kapitel 4.3.2). Die Dienstbeschreibung besteht im Kern, neben einer Funktionsbeschreibung, zum einen aus einer Menge an virtuellen Parametern und Parameterwerten, die bezeichnen, wann der Dienst gültig ist (*Precondition Parameter*) und zum anderen aus einer Menge an virtuellen Parametern, die der Dienst als Input benötigt (*Required Parameters*). Das ermöglicht, sowohl spielübergreifende Dienste anzubieten, wie beispielsweise einen Dienst zur externen Anzeige der Lokation im Spiel, der immer dann angezeigt wird, wenn der Spieler sich nicht im PvP-Modus befindet, genauso wie Dienste, die für ein Spiel spezialisiert sind, wie etwa ein Dienst, der dem Spieler innerhalb des Spiels zusätzliche Informationen anzeigt, sobald er im Spiel definierte Bereiche (Gebiete) wechselt. Die Verwendung der virtuellen Parameter als Referenz für die Eigenschaften von virtuell kontextbasierten Diensten bietet die Grundlage für eine formal definierte Dienstbeschreibung, die für die Dienstentwickler auf bekannten Konzepten basiert. Die Kombination aus *Precondition* und *Required Parametern* erlaubt eine Überprüfung eines Dienstes schon bei der Registrierung. So kann anhand der spielspezifischen Regelwerke (*Game Rule Sets*), die in der VCBS-Registry vorliegen, ein Mißbrauch der Daten ausgeschlossen werden, da sich aus der Dienstbeschreibung ableiten läßt, in welchen virtuellen Situationen welche Kontextinformationen an einen externen Dienst weitergegeben werden sollen.

Ein VCBS-Server verarbeitet die Kontextinformationen eines Benutzers und überprüft im **Dienst-Matching**, welche Dienste für den aktuellen Benutzerkontext gültig sind. Das Dienst-Matching wird anhand der Vorbedingungen der Dienste (*Precondition Parameter*) durchgeführt. Diese Vorbedingungen sind Bestandteil der Dienstbeschreibung und spezifizieren, in welchen virtuellen Situationen ein Dienst gültig ist. Der "Fisherman's Friend" Dienst ist beispielsweise nur in bestimmten Gebieten (an virtuellen Gewässern) gültig. Diese Gebiete können über die *Precondition Parameter* definiert werden. Der VCBS-Server kann mit Hilfe der xgdl, die die virtuellen Parameter des Charakters beinhaltet, seinen genauen Standort innerhalb der virtuellen Umgebung mit den Gebieten vergleichen, in denen der "Fisherman's Friend" Dienst gültig ist, und somit feststellen, ob sich der Charakter innerhalb eines in Frage kommenden Bereichs aufhält oder nicht. Analog verhält es sich für die Angelausrüstung, die eine weitere Vorbedingung des "Fisherman's Friend" Dienstes ist. Über die *Precondition Parameter* kann beispielswei-

se auch eine Liste definiert werden, die alle in Frage kommenden Gegenstände enthält und dann beim Dienst-Matching mit dem Gegenstand, den der Charakter gerade in der Hand hält, abgeglichen werden kann. Die Angabe der Kontextparameter, die ein Dienst benötigt (*Required Parameter*), um Informationen darzustellen oder Servicedaten zu berechnen, ist ein weiterer Bestandteil der Dienstbeschreibung.

6.1.3 Kommunikation zwischen den Komponenten

Um die zuvor beschriebenen Funktionen zu realisieren, sind bestimmte Kommunikationsabläufe zwischen den Komponenten der Middleware notwendig. Der Informationsaustausch beinhaltet die Übertragung der virtuellen Kontextinformationen (virtuelle Parameter). Wie in Kapitel 4.2.3 beschrieben, haben virtuelle Parameter unterschiedliche Eigenschaften, diese beeinflussen auch die Bestimmung der geeigneten Form der Übertragung. Zum einen müssen Parameter, die sich schnell ändern können, mit kurzen Update-Raten weitergegeben werden, zum anderen gibt es viele Parameter, die sich eher selten ändern, aber dafür sicher übertragen werden müssen.

Im Folgenden werden zunächst verschiedene Kommunikationstechnologien kurz eingeführt sowie die im Prototyp der VCBS-Middleware implementierte Kommunikationstechnik vorgestellt. Anschießend wird das für die Middleware entwickelte Kommunikationsprotokoll für die Kommunikation zwischen VCBS-Client und VCBS-Server vorgestellt.

Realisierung der Kommunikation

Es gibt verschiedene Möglichkeiten, die unterschiedlichen Anforderungen an die Übertragung zu erfüllen. Zum einen können verschiedene Protokolle und Kommunikationstechnologien verwendet werden, zum anderen kann mit einer Kommunikationstechnologie und unterschiedlichen Modi gearbeitet werden.

Auf Ebene der Transportschicht können verschiedene Protokolle (wie TCP oder UDP) für die Kommunikation verwendet werden. Es können bestehende Netzwerk-Bibliotheken, eine Kommunikations-Middleware oder eine Netzwerk-Engine eingesetzt werden. Beispiele dafür sind ENet [50], OpenTNL [29], RakNet [32] oder Zoidcom [49].

Auf der Applikationsschicht gibt es ebenso verschiedene Kommunikationstechnologien, die sich für die Übertragung der Informationen eignen. Dabei wird zwischen asynchronen und synchronen Technologien unterschieden.

Eine **Message Oriented Middleware** (MOM) realisiert eine indirekte asynchrone Kommunikation mittels der Verwendung von Warteschlagen (*Message-Queues*). Das ermöglicht eine lose Kopplung, da die beiden Kommunikationspartner nicht in direkter Verbindung miteinander stehen. Der Sender schickt Nachrichten, die er übermitteln möchte, an die Warteschlange des Empfängers (*Message Queuing*). Der Empfänger kann diese nun zu einem beliebigen Zeitpunkt abholen. Der Sender erhält jedoch keine Information über den Verbleib seiner Nachricht. Dies ermöglicht, dass Sender und Empfänger zu unterschiedlichen Zeiten verfügbar sind, da Nachrichten in der Warteschlange zwischengespeichert werden. *Message Queuing* realisiert eine Point-to-Point Kommunikation. Es gibt zwei weitere Kommunikationsmodelle auf Basis von MOMs, die Verteilung der Nachrichten eines Senders an mehrere Empfänger (*Publish/Subscribe*) und die Zwei-Wege-Kommunikation (*Request/Reply*). Bei der Verwendung von *Publish/Subscribe* "publiziert" der Sender seine Nachricht zu einem bestimmten Topic und die Interessenten (*subscriber*) registrieren sich für dieses Topic. Die Nachrichten werden dann über einen Message-Broker übertragen und es gibt keine Bestätigungen, ob eine Datenübertragung erfolgreich war. Ein Beispiel hierfür sind RSS-Feeds. Zur Realisierung der *Request/Reply* Kommunikation wird mit der asynchronen Kommunikationsinfrastruktur eine synchrone Kommunikation nachgebildet. Die beiden Kommunikationsrichtungen werden durch zwei Warteschlangen (Request-Queue und Reply-Queue) abgebildet. Eine weit verbreitet Java-API für MOMs ist der Java Messaging Service (JMS) [55]. JMS-Nachrichten können

neben einfachem Text auch Key/Value-Paare oder serialisierte Objekte enthalten. Somit lassen sich damit auch XML-Dokumente ohne Weiteres übertragen. JMS kann jedoch nicht über HTTP betrieben werden und muss daher beispielsweise getunnelt werden. Für Daten, die zuverlässig übertragen werden müssen (Sender erhält eine Empfangsbestätigung des Receivers), sind *Message Queuing* und *Publish/Subscribe* basierte Kommunikationsverfahren nicht geeignet.

Web Services realisieren ein synchrones Kommunikationsverfahren mit *Request/Response* Interaktion. Bei Web Services gibt es immer einen Kommunikations-Server, der über eine URL Kommunikationsanfragen des Kommunikations-Clients entgegennimmt. Der Kommunikations-Client erhält eine Bestätigung über eine erfolgreiche Datenübertragung. Web Services sind offen und flexibel, sie können auf verschiedene Übertragungsprotokolle aufsetzten und verwenden bestehende Standards (wie HTTP oder XML). Es gibt verschiedene Ausprägungen bei der Verwendung von Web Services, wie SOAP [74] und REST (*Representional State Transfer*) [26]. Bei Verwendung von SOAP werden die Daten in XML gekapselt und die Kommunikation basiert auf HTTP. Der Nachteil ist, dass dadurch ein vergleichsweise großer Overhead entsteht. Zwei verbreitete Implementierungen von SOAP für Java sind Apache Axis (Apache eXtensible Interaction System) [3] und JAX-WS [56].

Da die VCBS-Middleware im Rahmen dieser Arbeit prototypisch implementiert wurde, um die Realisierbarkeit der Konzepte nachzuweisen, wird eine etablierte Kommunikationstechnologie eingesetzt. Die Implementierung erfolgte unter Verwendung von Web Services auf Basis von JAX-WS mittels SOAP und HTTP. Trotz des damit entstehenden Overheads bei der Verwendung von Web Services gegenüber anderen Kommunikationstechnologien sind die synchrone Kommunikation und die Nutzung von HTTP ausschlaggebend für die Auswahl gewesen.

Kommunikationsprotokoll

Im Folgenden wird beispielhaft das entwickelten Kommunikationsprotokoll im Detail anhand der Kommunikation zwischen VCBS-Client und VCBS-Server dargestellt. Für eine Entlastung der Kommunikation werden zwei verschiedene Modi eingesetzt, um die erfassten virtuellen Kontextinformationen vom VCBS-Client an den VCBS-Server zu übertragen. Zum einen gibt es die Möglichkeit alle erfassten Informationen gebündelt als *IngameInformationObject* (IIO) zu übertragen, zum zweiten können auch mehrere virtuelle Parameter (VP) separat übertragen werden. Eine Übertragung der gesamten Parameter findet statt, wenn sich genügend Parameter geändert haben. Der *Threshold* für die Anzahl der Parameter kann in der Middleware konfiguriert werden.

Abbildung 6.4 zeigt den Kommunikationsablauf zwischen dem VCBS-Client und dem VCBS-Server. Die dargestellte Kommunikation findet statt, wenn sich ein Benutzer bei der Registry angemeldet hat und von dieser einem VCBS-Server zugewiesen wurde. Die initiale Informationsübertragung des VCBS-Clients an den VCBS-Server beinhaltet ein vollständiges Set (IIO) der virtuellen Kontextinformationen des Benutzers, sodass der VCBS-Server die virtuelle Situation des Benutzers ableiten kann. Auch bei großen Kontextänderungen, die über dem *Threshold* liegen, werden die Kontextinformationen vollständig übertragen - (1) *transmitIIO*. In einem Kommunikations-Loop fragt der Client den Server regelmäßig nach neuen Dienstangeboten - (2) *getServiceOffers*. Diese werden dann in Form von *ServiceOffers* an den Client übertragen und für den Benutzer in das Spiel integriert. Hat der Benutzer mindestens einen Dienst aktiviert, so fragt der Client zusätzlich neue Servicedaten ab - (3) *getServiceData*. Die von einem Dienst für den Benutzer bereitgestellten Daten, werden beim Server gesammelt und als *ServiceDataObjects* zum Client übertragen und von diesem eingebettet. Dies geschieht bei allen Anfragen, außer der Angebotsanfrage, wenn Daten vorhanden sind. Aktiviert oder deaktiviert der Benutzer einen Dienst, so wird dies als *Activation*-Nachricht an den Server weitergegeben - (4) *serviceActivation*. Bei kleinen Kontextänderungen können die geänderten virtuellen Parameter als *VirtuelParameterObject* übertragen werden - (5) *transmittVP*. Die Daten, die der Benutzer als Eingabe für einen Dienst erzeugt, werden als *ClientDataObjects* an den Server weitergegeben - (6) *sendClientData*.

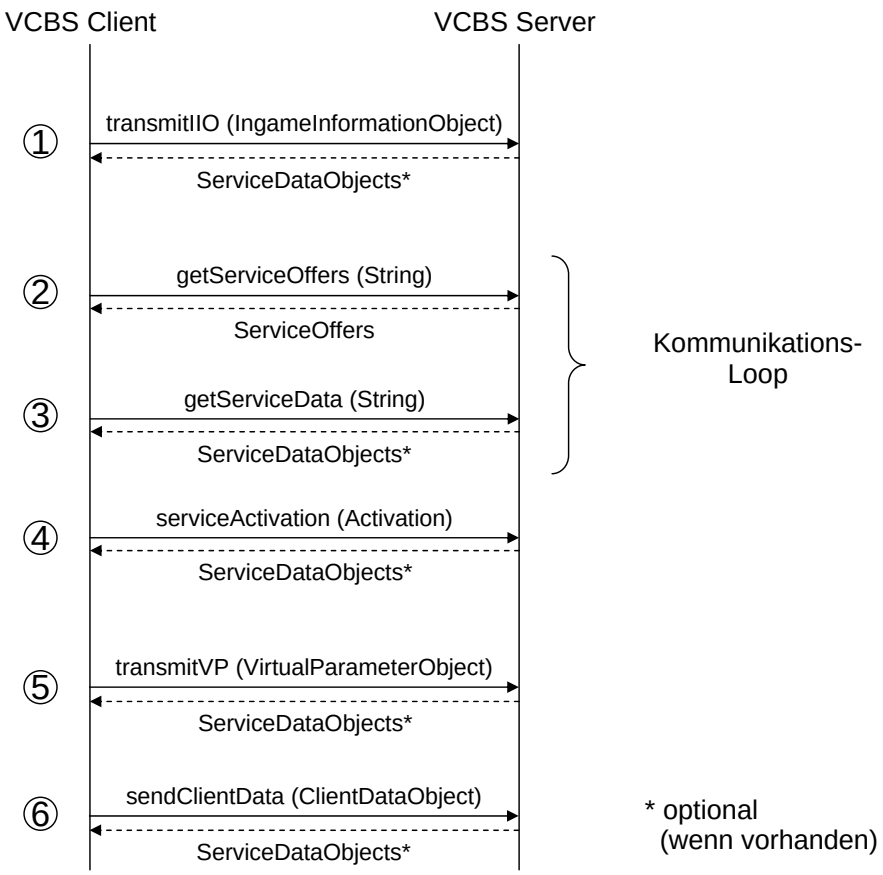

Abbildung 6.4: Kommunikation zwischen VCBS-Client und VCBS-Server

6.2 Anbindung von Spielen mittels Plug-Ins

In den aktuell von der Industrie angebotenen MOGs ist eine gemeinsame generische Spieleschnittstelle, wie sie auf Basis der xgdl umgesetzt werden kann, nicht integriert. Aus diesem Grund wird in dieser Arbeit die Verwendung von Plug-Ins zur Anbindung von Spielen umgesetzt. Die Vorteile der Plug-In Technologie sind große Flexibilität und geringer Erweiterungsaufwand.

Im Folgenden werden verschiedene Aspekte der Anbindung von Spielen an die VCBS-Middleware durch Plug-Ins vorgestellt und im Weiteren die in dieser Arbeit entwickelten Spiele Plug-Ins für *World of Warcraft* [11] und *Ultima Online* [47] kurz vorgestellt.

6.2.1 Formen der Anbindung von Spielen durch Plug-Ins

Für die Interprozesskommunikation zwischen dem Spiel-Client und dem Spiel Plug-In im VCBS-Client muss eine geeignete Schnittstelle geschaffen werden. Für den Informationsaustausch zwischen dem VCBS-Client und einem Spiel stehen grundsätzlich verschiedene Technologien zur Verfügung. Die Kommunikation zwischen Spiel und VCBS-Client erfolgt lokal auf dem System des Benutzers. Entsprechende Technologien sind die Verwendung von *Sockets*, *Pipes*, *Shared Memory* oder von *Dateien*. Bei der Verwendung von **Shared Memory** verwenden beide Teilnehmer einen gemeinsamen Speicherbereich, um Daten auszutauschen. Daten, die von einem Teilnehmer in den Speicher geschrieben werden, können dann vom anderen Teilnehmer ausgelesen werden. Das Verfahren ist für den Einsatz bei Spielen jedoch nicht geeignet, da Spiele aus Sicherheitsgründen (Cheating) keine fremden Speicherzugriffe erlauben. Mit lokalen Netzwerk-**Sockets** kann eine bidirektionale Datenübertragung realisiert werden. Die Daten-

übertragung kann durch einen Datenstrom (*Stream Sockets*) oder über einzelne Nachrichten (*Datagram Sockets*) erfolgen. Übliche Protokolle sind dabei TCP für *Stream Sockets* und UDP für *Datagram Sockets*. Wird beispielsweise TCP verwendet, so kann ein Spiel einen lokalen *Socket* öffnen, wobei der entsprechende TCP-Port in der Konfiguration des Plug-Ins im VCBS-Client hinterlegt ist. Der VCBS-Client kann sich nun zu dem *Socket* verbinden und mit dem Spiel Daten austauschen. Neben den *Stream Sockets* bieten auch **Pipes** eine Kommunikation über Datenstöme. Um Daten über eine *Pipe* oder auch eine *named Pipe* zu versenden, öffnet ein Teilnehmer zunächst eine *Pipe* und sendet dann Daten an diese. Nun kann der andere Teilnehmer aus dieser *Pipe* die Daten lesen. Die Verwendung von *Sockets* ist jedoch, im Gegensatz zu *Pipes*, bei MOGs in der Regel bereits integriert, da *Sockets* auch für spielinterne Netzwerkkommunikation zwischen Spiel-Client und Spiel-Server verwendet werden. Die Verwendung von **Dateien** zum Informationsaustausch erfolgt, indem ein Teilnehmer Daten in eine Datei schreibt und der andere Teilnehmer dann diese Daten wieder aus der Datei liest. Vorteile, die bei der Verwendung von Dateien entstehen, sind die Speicherung der Daten in der Datei und dass eine Protokollierung von Spielständen in Log-Dateien in der Regel bei allen Spielen durchgeführt wird, sodass die grundlegenden Mechanismen hierfür schon in den Spielen integriert sind. Am besten für den Datenaustausch geeignet, da bereits Teil des Spiel-Clients, sind somit *Sockets* und der Austausch von Dateien.

Die Interprozesskommunikation, die für ein Spiel eingesetzt werden kann, hängt von den Voraussetzungen des Spiels ab. Bietet ein Spiel die Möglichkeit einer Interprozesskommunikation mittels *Sockets*, so kann ein synchroner Datenaustausch zwischen Spiel und VCBS-Client realisiert werden. Ist diese Möglichkeit nicht gegeben, so sind austauschbare Dateien eine Alternative, wobei damit oft nur ein asynchroner Datenaustausch realisiert werden kann, da ein Zugriff auf diese Dateien normalerweise nicht zu jedem Zeitpunkt möglich ist.

Darüber hinaus gibt es noch eine weitere wichtige Möglichkeit, die im Rahmen des Datenaustauschs zwischen Spiel-Client und Spiel Plug-In eingesetzt werden kann. Mithilfe der Skriptsprache Lua [30] lassen sich Informationen aus einem Spiel abfragen und weiter aufbereiten. Lua wurde 1993 von der Tecgraf (Computer Graphics Technology Group) an der PUC-Rio (Pontifical Catholic University of Rio de Janeiro) in Brasilien entwickelt [33]. Lua wird bei Computerspielen eingesetzt, um einzelne Komponenten eines Spiels (wie Konfigurationsdateien) von der Spiele-Engine zu trennen. Vor allem Aufgrund der geringen Größe des Interpreters ist Lua für Spiele interessant und wird dort weit verbreitet eingesetzt, Beispiele dafür sind: *World of Warcraft*, *SimCity4*, *Warhammer Online: Age of Reckoning*, *FarCry*, *Heroes of Might and Magic V*, *S.T.A.L.K.E.R.: Shadow of Chernobyl*, *Homeworld 2*, oder *SpellForce - The Order of Dawn* [34]. Aber auch in anderen Anwendungen wird Lua eingesetzt, Beispiele dafür sind: Adobe Photoshop Lightroom, LuaTeX, MySQL Proxy, oder Wireshark [34].

Die Skriptsprache Lua hat verschiedene Vorteile. Sie ist flexibel, kann sowohl eigenständig als auch eingebettet in andere Programme verwendet werden. Sie ist schnell (im Vergleich zu anderen Skiptsprachen), erprobt und robust (was ihr Einsatz in vielen kommerziellen Programmen zeigt). Sie ist die in Spielen am häufigsten verwendete Skriptsprache und und frei verfügbar (MIT Lizenz).

Die Interprozesskommunikation vom Spiel zum VCBS-Client ist auf die in-game Informationen beschränkt, die das Spiel bereitstellt. Durch den Einsatz von Lua kann jedoch die Abfrage von in-game Informationen gezielt durchgeführt werden. Dadurch können die benötigten virtuellen Parametern abgefragt und dem VCBS-Client direkt im xgdl-Format bereitgestellt werden. Darüber hinaus kann durch die Verwendung von Lua das Service User Interface einfach in das Benutzerinterface des Spiels integriert werden. Aus diesem Grund wird Lua auf der Spiel-Seite des Plug-Ins eingesetzt, falls ein Spiel diese Möglichkeit bietet.

6.2.2 Entwickelte Spiele Plug-Ins

Im Rahmen der Umsetzung der VCBS-Middleware wurden zwei Spiele Plug-Ins entwickelt. Spiele, welche die VCBS-Middleware verwenden möchten, müssen neben den technischen Voraussetzung, also einer

Anbindung des Spiels an den VCBS-Client durch ein entsprechendes Plug-In, auch die rechtlichen Voraussetzung erfüllen. Spiele, die einen Zugriff auf in-game Informationen verbieten, können aus rechtlichen Gründen nicht mit der VCBS-Middleware verwendet werden. Auch Darstellung und Nutzung des Service User Interfaces innerhalb der Spiele muss möglich sein.

Das Spiel *World of Warcraft* wurde aufgrund seiner Marktdominanz und der gegebenen Möglichkeiten, eigene Erweiterungen, sogenannte Addons, lokal in das Spiel einzubinden, schon mehrfach in dieser Arbeit betrachtet. Aufgrund des hohen Stellenwertes von *World of Warcraft* wurde ein *World of Warcraft* Plug-In implementiert. Dieses wird in der Evaluation der VCBS-Middleware eingesetzt (siehe Kapitel 7). Das MMORPG *Ultima Online* wurde bereits 1997 veröffentlicht und wird heute als wegweisend im Bereich der MMOGs bezeichnet [28]. Die Basis, auf der Ultima Online aufbaut, ist auch heute noch Grundlage für die Entwicklung neuer MMOGs. Aus diesem Grund wurde als zweites ein Plug-In für *Ultima Online* implementiert.

World of Warcraft

Die Addons, die für *World of Warcraft* erstellt werden, dienen hauptsächlich dazu, die Benutzeroberfläche besser an den Spieler anzupassen, Abläufe zu vereinfachen und weitere Informationen anzuzeigen. Durch die Implementierung eines entsprechenden Addons können auch die für eine Anbindung des Spiel-Clients an den VCBS-Client benötigten Funktionen, wie Datenaustausch und Integration des Service User Interfaces, umgesetzt werden. Daher wurde im Rahmen dieser Arbeit beispielhaft ein entsprechendes *World of Warcraft* Addon entwickelt.

Funktionen und Layout der Benutzeroberfläche von *World of Warcraft* werden durch Lua- und XML-Dateien definiert. Für die Darstellung von Elementen der graphischen Benutzeroberfläche wird eine deklarative XML-Syntax verwendet. Die verschiedenen Elemente des Interfaces werden in jeweils eigenen XML-Dateien definiert. Die Funktionen das Interfaces werden durch Lua-Scripte zur Verfügung gestellt. Die Lua-Scripte können entweder in den XML-Dateien integriert werden oder in separate Lua-Dateien ausgelagert werden. Die Lua-Funktionen werden durch entsprechende Ereignisse, die im Spiel auftreten, getriggert. Der Vorteil für das Spiel besteht darin, dass die Darstellung für den einzelnen Benutzer entkoppelt wird und eine Benutzeranpassung möglich wird. So ist beispielsweise das Betreten des Servers (*Realms*) mit dem virtuellen Charakter ein spezielles Ereignis. Basierend darauf wird dann das primäre Interaktionsfenster das Service User Interfaces nach dem Betreten des *Realms* angezeigt und ermöglicht die Interaktion mit den verfügbaren Diensten. Im Spielverlauf auftretende Ereignisse werden in Form von *Events* an das Interface weitergegeben, die wiederum die Grundlage für das Auslösen von bestimmten Funktionen sein können.

Das Addon "Marker2" sammelt Daten des Spiels und protokolliert sie in einer Datei. Das Lua-Addon umfasst auch die Bereitstellung des Service User Interfaces innerhalb des Spiels und die entsprechenden Interaktionselemente. Das entwickelte Addon besteht aus mehreren XML- und Lua-Dateien, sowie einer TOC-Datei, die Metadaten zum Addon enthält und alle XML- und Lua-Dateien auflistet, welche beim Betreten eines *Realms* geladen werden. Weitere Lua-Dateien werden aus dem XML-Markup oder den Lua-Dateien selbst eingebunden. Hinzu kommt noch eine zusätzliche Bibliothek, die das Scheduling von Tasks unterstützt (*Chronos*) [14].

Die einzigen Beschränkungen, die sich bei der Verwendung des Addons ergeben, sind die rechtlichen Bestimmungen, die in der *World of Warcraft* EULA (End User License Agreement) spezifiziert sind. Der Datenaustausch mit dem Spiel kann in diesem Fall nicht synchron über Netzwerk-*Sockets* oder Dateien erfolgen. Zusätzlich wurden die entsprechenden Lua-Funktionen in der von *World of Warcraft* verwendeten Version von Lua entfernt, um Cheating vorzubeugen. Es besteht jedoch die Möglichkeit, Daten sitzungsübergreifend in einer persistente Variable zu speichern. Mittels Hashes in Lua und Arrays lassen sich so auch komplexe Datenstrukturen abbilden. Der Zugriff auf die in-game Daten ist daher erst nach dem Verlassen des Servers möglich. Die so gespeicherten Informationen werden dann vom VCBS-Client ausgelesen und vom *World of Warcraft* Plug-In in die xgdl-Datenstruktur konvertiert. Alle benötigten Basisinformationen (Basis-xgdl) können so erfasst werden.

Ultima Online

Für die Schnittstelle zwischen *Ultima Online* und dem VCBS-Client wurde eine entsprechende Lua-Erweiterung des *Ultima Online* Clients entwickelt. Die Lua-Erweiterung ist analog zum *World of Warcraft* Addon für die Darstellung und Interaktion mit dem Service User Interface zuständig, sowie für die Sammlung der in-game Informationen. Darüber hinaus werden die in-game Informationen aktiv an den VCBS-Client weitergegeben, sowie Servicedaten vom VCBS-Client entgegengenommen und über das Service User Interface dargestellt. Die in-game Informationen, die während des Aufenthalts eines virtuellen Charakters in der virtuellen Welt entstehen, werden erfasst und an das *Ultima Online* Plug-In des VCBS-Clients weitergegeben. Die Weitergabe der in-game Informationen erfolgt über über TCP-*Sockets*. Der realisierte Informationsaustausch ist bidirektional, die von den verwendeten Diensten erzeugten Servicedaten werden vom VCBS-Client über den TCP-*Socket* übertragen und durch die Lua-Erweiterung im Spiel dargestellt.

6.3 Realisierte Dienste

Es gibt verschiedene Arten von Diensten, die zur Unterstützung von Spielern genutzt werden können (siehe Kapitel 4.1.4). Typische Dienste sind Kommunikations-, Kooperations-, Informations-, Dokumentationsdienste oder Kombinationen daraus (siehe auch Kapitel 5.3.3).

Es gibt auch Dienste, die bereits von den Spielerhersteller innerhalb von Spielen bereitgestellt werden. Dazu zählen Kooperationsdienste, beispielsweise zur Unterstützung des Handels, wie etwa ein spezielles Handelsinterface, ein für den Handel bestimmter Text-Chat, ein in-game Postsystem oder eine spielinterne Auktionsplattform. Weitere Dienste innerhalb von Spielen sind Informationsdienste, wie die Minimap, oder auch kombinierte Dienste, wie Statusanzeigen von Gruppenmitgliedern. Diese Dienste erfüllen alle Basisfunktionalitäten, jedoch decken sie nicht alle Bedürfnisse der Benutzer ab.

Die in dieser Arbeit entwickelten Beispieldienste umfassen verschiedene Dienstarten und bieten erweiterte Funktionalitäten, die nicht durch die spielinterne Basisdienste abgedeckt werden. Im Folgenden werden die implementierten Beispieldienste der "**dynamische Sprachkommunikationsdienst**" (dSK-Dienst), der "**verteilte Ping**", der "**Fisherman's Friend**" und der "**lokationsbasierte Annotationsdienst**" (LBA-Dienst) vorgestellt. In Kapitel 7.2 werden zwei dieser Dienste für die Evaluation der Middleware-Funktionalitäten verwendet. Dabei handelt es sich zum einen um den "Fisherman's Friend" Dienst, der schon mehrfach in Verlauf der Arbeit als Beispieldienst verwendet wurde, da er alle Funktionalitäten der VCBS-Middleware nutzt, und zum anderen um den LBA-Dienst.

6.3.1 Dynamische Sprachkommunikation

Um die Interaktion in einer Gruppe zu unterstützen, hat sich die Verwendung von Sprachkommunikation zusätzlich zum Spielen von MOGs etabliert. Die Sprachkommunikation bietet, gegenüber der traditionellen textbasierten Kommunikation, zwei wichtige Vorteile. Zum einen ist Sprachkommunikation schneller (in der Regel können Menschen schneller reden als schreiben) und zum anderen kollidiert sie nicht mit der Steuerung des Spiels (für beides, Schreiben sowie Spielsteuerung werden die Hände und meisten auch dasselbe Eingabegerät - die Tastatur - benötigt, Sprechen benötigt diese nicht). Heutzutage werden in den meisten Fällen spezielle spielexterne Tools für die Sprachkommunikation verwendet.

Diese **Sprachkommunikations-Tools** (*Voice-Chats*) bieten statische Sprachkanäle, die eine gewisse Anzahl von Kommunikationsteilnehmern unterstützen. Eine Gruppe teilt sich einen gemeinsamen Kommunikationskanal (unabhängig von der Gruppengröße) wobei immer nur ein Gruppenmitglied sprechen kann, da das Gesagte sonst unverständlich wird, egal ob es sich um eine 5er Gruppe oder um eine 40er Gruppe handelt. Diese Tools müssen vor Beginn eines Spielabschnittes gestartet und eingerichtet werden, da sie eine eigenständige Applikation darstellen, die nicht mit dem Spiel verknüpft ist. Um eine

Umkonfiguration (Kanal- oder Serverwechsel) durchzuführen, muss der Spieler erst das Spielinterface verlassen und zu der Applikation des Voice-Chats wechseln.

Der *dynamische Sprachkommunikationsdienst* (dSK-Dienst) kann den Voice-Chat umkonfigurieren, ohne dass der Benutzer aktiv werden muss. Basierend auf den Kontextinformationen seiner Benutzer, kann der Dienst die Benutzer zu bestimmten Kommunikationsservern verbinden oder zwischen Kommunikationskanälen wechseln lassen.

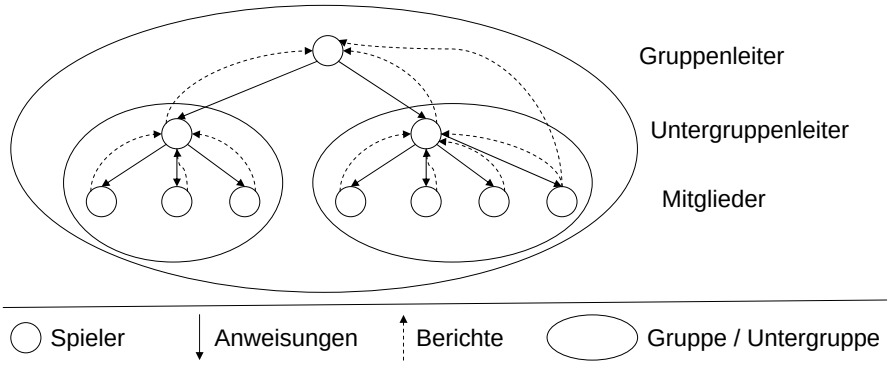

Abbildung 6.5: Aufbau von Spielergruppen

Abbildung 6.5 stellt beispielhaft den Aufbau von Spielergruppen dar. Strukturen von Gruppen (Teamstrukturen) in Spielen können wie gezeigt sehr komplex sein (siehe Kapitel 2.1 und 4.1.4). Die dargestellte Teamstruktur besteht aus drei Ebenen, einem Gruppenleiter (*Teamleader*) und zwei Untergruppen mit entsprechend jeweils einem Untergruppenleiter und den eigentlichen Gruppenmitgliedern. Eine solche Struktur findet sich sehr häufig in Spielen, in denen Gruppen mit mehreren Spielern benötigt werden, vor allem, wenn Rollen von mehr als einem Spieler ausgeführt werden. Dann entstehen oft Situationen, in denen nicht nur eine Aufgabenteilung nötig ist, sondern auch mehrere Spieler dieselbe Aufgabe ausführen, was wiederum eine Absprache der entstehenden Untergruppen nötig macht. Befinden sich jedoch alle Spieler in nur einem Kommunikationskanal, dann stören sich die Absprachen der einzelnen Untergruppen gegenseitig und der Teamleader, der in einer schwierigen oder brenzligen Situation die Möglichkeit haben muss, alle Mitglieder zu koordinierten, kommt dann im schlimmsten Fall gar nicht mehr zu Wort. Das kann dann schnell das Scheitern der Gruppe herbeiführen. Je mehr Mitglieder eine Gruppe hat und je komplexer die zu lösenden Aufgaben werden, desto größer wird auch der Kommunikationsbedarf.

Bei Verwendung des dSK-Dienstes kann die Kommunikationsstruktur der aktuellen Spielsituation angepasst werden. Abbildung 6.6 zeigt verschiedene situationsabhängige Verteilungen der Spieler in Kommunikationskanäle:

a) Die Basissituation: alle Spieler befinden sich in einem Kommunikationskanal. Diese Kommunikationsstruktur bildet die Standardaufteilung, vor allem vor und nach dem eigentlichen Spielen wird diese Aufteilung für Planung, Taktikbesprechung oder Analyse verwendet.

b) Situation 1: es werden drei Kommunikationskanäle verwendet. Die beiden Untergruppen *B* und *C* haben jeweils einen eigenen Kommunikationskanal, in dem sie sich untereinander absprechen können. Die drei Teamleader haben noch einen separaten Kommunikationkanal, den "Teamleader-Kanal" *A*.

c) Situation 2: es werden wiederum drei Kommunikationskanäle verwendet, jedoch in einer anderen Aufteilung. Neben dem Teamleader-Kanal ist die Gruppenaufteilung eine andere. Der Teamleader ist diesmal aktives Mitglied der Gruppe *C*, dafür unterstützt *Spieler Nr. 6* diesesmal Gruppe *B*.

Der dSK-Dienst verwendet den virtuellen Kontext der Spieler um die Kommunikationssituation der Spielsituation anzupassen. Durch den dSK-Dienst können beispielsweise die Mitglieder virtueller Gruppen automatisch in denselben Kommunikationskanal verbunden werden. Die Verwendung mehrerer dynamisch wechselbarer Kommunikationskanäle ermöglicht eine gleichzeitige Kommunikation in verschiedenen Untergruppen, die in einem normalen einfachen Kommunikationskanal nicht möglich ist. Mittels

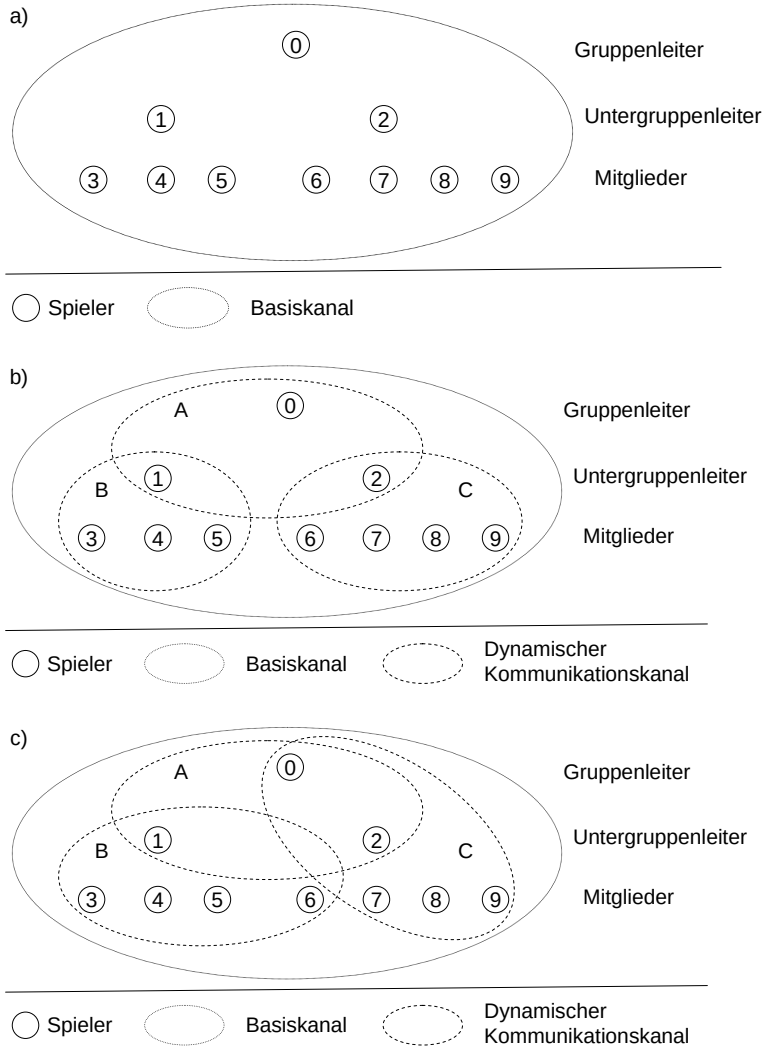

Abbildung 6.6: Situationsabhängige Verteilung von Spielern in Kommunikationskanäle: a) Basissituation, b) Situation 1, c) Situation 2

des dSK-Dienstes können die dynamischen Kanäle basierend auf dem virtuellen Kontext der Spieler verwaltet und der jeweiligen Spielsituation angepasst werden. Zusätzlich bietet er die Möglichkeit einer lokationsbezogenen Klang- und Laustärkenanpassung in einer virtuellen Umgebung, zum Beispiel um realistischere Kommunikationsszenarien in Rollenspielen zu ermöglichen (siehe auch [SA04]).

Die Implementierung des dSK-Dienstes verwendet das Open-Source Sprachkommunikations-Tool Mumble ([42]). Dessen Kommunikations-Server (Murmur) erlaubt die gleichzeitige Präsenz eines Benutzers in verschiedenen Kanälen (mit unterschiedlichen Rechten) und kann über den dSK-Dienst gesteuert werden. Dadurch kann die angebotene Kommunikationsstruktur an die jeweilige Spielsituation angepasst werden, was eine situationsbezogene Kommunikation ermöglicht.

6.3.2 Verteilter Ping

Ein weiterer Dienst zur Unterstützung der Gruppenkooperation ist der *verteilte Ping*. Dieser Dienst unterstützt das koordinierte Ausführen von Aktionen in Spielen. Dafür wird ein Audiosignal auf verteilten Clients zeitsynchron abgespielt. Die synchrone Abstimmung ist wichtig um gemeinsame Aktionen ausführen zu können. Vor allem im PvP (Spieler gegen Spieler) ist die genaue Abstimmung verschiedener

Einzelaktionen ein bedeutender Faktor. Im Spiel *GuildWars* [5] gibt es beispielsweise kombinierte Angriffe, die in weniger als einer Minute abgeschlossen werden müssen, um erfolgreich zu sein. Der "verteile Ping"-Dienst bietet eine zeit-synchrone Audio-Benachrichtigung, die von einem Gruppenmitglied getriggert werden kann und unter Berücksichtigung der unterschiedlichen Latenzen der Gruppenmitglieder frei konfigurierbare Audiosignale zeitgleich bei allen Gruppenmitgliedern abspielt. Der verteilte Ping benutzt dafür ein NTP (Network Time Protocol [40]) ähnliches Protokoll und Latenzmessungen zu den verschiedenen Clients der Gruppenmitglieder.

Die Benutzung des "verteile Ping"-Dienstes kann manuell getriggert werden. Darüber hinaus besteht die Möglichkeit den Dienst basierend auf virtuellen Kontextinformationen zu triggern, diese müssen jedoch zuerst im Dienst definiert werden.

6.3.3 "Fisherman's Friend"

Der "*Fisherman's Friend*" ist ein Informationsdienst, der Zusatzinformationen für das Angeln in virtuellen Umgebungen bereitstellt. Während ein Benutzer in der virtuellen Umgebung Angeln geht, liefert ihm der "Fisherman's Friend" Dienst nützliche Zusatzinformationen.

Angeln in virtuellen Umgebungen ist normalerweise als Teil des *Harvesting*-Systems (Ernte-System) einer virtuellen Welt definiert. Daher muss ein virtueller Charakter, um Angeln zu können, eine Angelfertigkeit besitzen, einen zum Angeln geeigneten Gegenstand (meist eine Angelrute) in der Hand halten, an einem für das Angeln geeigneten Gewässer stehen und die Angel-Aktion aktivieren. Die virtuelle Welt berechnet dann, basierend auf den Attributen des virtuellen Charakters wie seiner Angelfertigkeit das Resultat der Angelaktion. Je nach Angelposition wird dabei aus der im *Harvesting*-System festgelegten Beute (*Loot*) eine Auswahl ausgewürfelt. Die Beute kann auch leer sein. Die so gefangene Beute erhält dann der virtuelle Charakter des Spielers in sein Inventar. Spieltheoretisch betrachtet, existieren *Payoff*-Matrizen, die Mengenangaben bzw. Erfolgswahrscheinlichkeiten enthalten. Eine höhere Angel-Fertigkeit oder eine bessere Angel kann dabei den *Payoff* positiv beeinflussen. Wirft ein Spieler seine virtuelle Angel aus, so wird im Hintergrund zufällig, basierend auf den gegebenen Wahrscheinlichkeiten, das Resultat bestimmt.

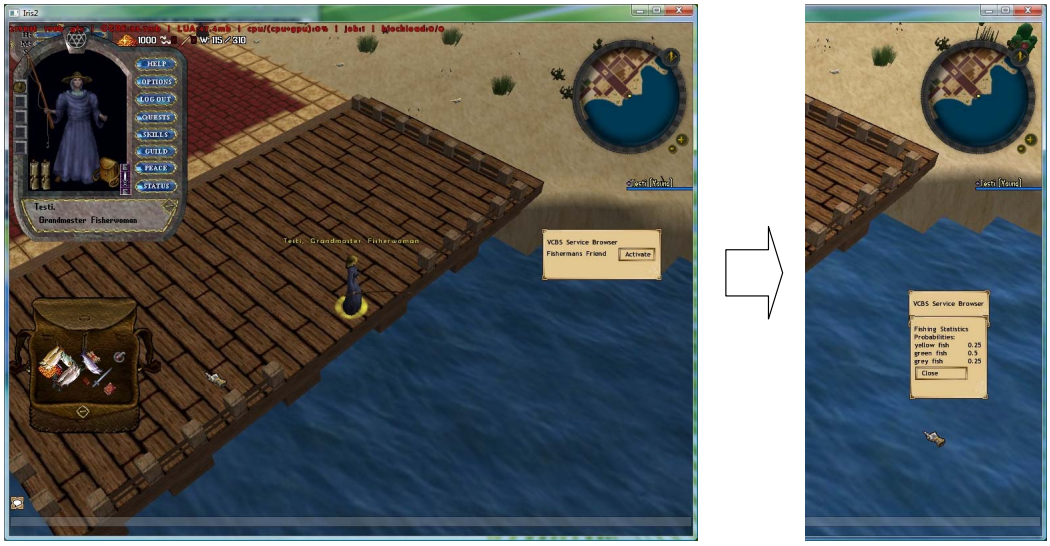

Abbildung 6.7: "Fisherman's Friend" Dienst zur Unterstützung des Angelns in virtuellen Welten

Wenn sich der virtuelle Charakter eines Benutzers an einem virtuellen See oder Fluss befindet und eine Angel in die Hand nimmt, dann wird ihm der "Fisherman's Friend" Dienst angeboten (siehe Abbildung 6.7 linke Seite). Aktiviert der Benutzer den Dienst, so werden seine Kontextinformationen gesammelt,

sowie die Daten über die von ihm gefangenen Fische. Letztere werden vom Dienst dazu verwendet, für den Angelplatz, an dem sich der virtuelle Charakter befindet, zu berechnen mit welcher Wahrscheinlichkeit dort verschiedene Fische geangelt werden können. Für die Ermittlung der Statistiken werden die Angelinformationen von allen Benutzern gesammelt, während sie in der virtuellen Welt angeln gehen. Dadurch hat der Dienst auch noch einen kollaborativen Aspekt, da die erzeugten Informationen verbessert werden, je mehr Benutzer den Dienst verwenden.

Die ermittelten Informationen werden dem Benutzer wiederum zur Verfügung gestellt. Weitere Zusatzinformationen, die basierend auf den Kontextinformationen des Benutzers bereitgestellt werden, sind zum Beispiel, ob die Schwierigkeitstufe dieser Stelle für den Charakter geeignet ist oder Informationen über andere Angelplätze, die noch für den Benutzer interessant sein könnten (z.B. basierend auf seinen Angelfertigkeiten). Diese Informationen werden nicht von der virtuellen Welt selbst zur Verfügung gestellt, können aber für den Benutzer sehr nützlich sein. Neben der in-game Bereitstellung (siehe Abbildung 6.7 rechte Seite), kann eine Internet-Applikation, die einen "Fisherman's Friend" anbietet, die ermittelten Informationen weiterverwenden und beispielsweise Fisch-Statistiken von verschiedenen Angelplätzen im Internet zur Verfügung stellen (siehe auch Abbildung 5.10).

6.3.4 Lokationsbasierte Annotation

Virtuelle Welten von MOGs können eine sehr große Fläche umfassen. Zum Beispiel bietet "World of Warcraft" eine riesige Spielwelt bestehend aus zwei Kontinenten (in der Basisversion), die wiederum in etliche Gebiete unterteilt sind, die aus verschiedenen Region bestehen und zusätzlich spezielle Bereiche (Instanzen) enthalten können. Gebiete können so groß ausfallen, dass diese sich "zu Fuß" kaum in 10 Minuten Echtzeit durchqueren lassen. Es gibt also große Flächen virtuellen Raums, die von den Spielern erkundet werden können. Spieler oder sogar Gruppen von Spielern verbringen daher viel Zeit mit der Erkundung der virtuellen Umgebung und der Erschließung von bestimmten Arealen oder Gebieten der virtuellen Welt. Für eine Dokumentation der Ergebnisse der Erkundungen werden häufig Wikis oder andere Dokumentations-Tools verwendet. So lassen sich die Fortschritte mit Team-Mitgliedern und der Community teilen. Die Dokumentation findet dabei außerhalb des Spieles statt und wird dadurch meist erst nach dem eigentlichen Erkunden vorgenommen. Daher ist die Angabe einer Referenz zu den verschiedenen Orten nicht immer leicht. Der Spieler kann sich beispielsweise Koordinaten notieren, auf die er später referenziert. Jedoch werden von vielen virtuellen Welten keine Postionskoordinaten bereitgestellt (oder sie müssen über Erweiterungen erst ermittelt werden).

Mit dem *lokationsbasierten Annnotationsdienst* (LBA-Dienst), können direkt im Spiel Annotationen eines Benutzers erfasst und dargestellt werden. Abbildung 6.8 zeigt den LBA "Marker Manager" mit dem Bedienelement "*Add Marker*" zum Erstellen neuer Annotationen.

Das primäre Interaktionsfenster "Marker Manager" ist Teil des Service User Interfaces und wird bei Verwendung des LBA-Dienstes angezeigt. Es bietet Zugriff auf bestehende Annotation und die Option, neue Annotationen zu erstellen. Annotationen können mit einem Namen sowie einer Beschreibung versehen werden. Die so erstellten Annotationen werden automatisch mit dem Lokationskontext des Benutzers verknüpft. Dafür werden alle verfügbaren Informationen über den aktuellen Standort des virtuellen Charakters einbezogen (Kontinent, Gebiet und Positionskoordinaten). Bei der Annäherung des virtuellen Charakters an eine Annotation wird diese wiederum automatisch angezeigt.

Der LBA-Dienst speichert die Annotationen der Benutzer, die diesen Dienst verwenden. Er bietet zusätzlich die Möglichkeit, die eigenen Annotationen mit anderen Benutzer zu teilen. Dabei ist es nicht nur möglich, mit in-game *Buddies* des Benutzers gemeinsam an Annotationen zu arbeiten. Es können auch durch die Internet-Applikation, die den Dienst bereitstellt, Annotationen weiteren Benutzergruppen zugänglich gemacht werden (beispielsweise mittels einer Community-Plattform). Darüber hinaus können die lokationsbezogenen Annotationen über die externe Internet-Applikation für die Community im Internet veröffentlicht werden.

Abbildung 6.8: Lokationsbasierter Annotationsdienst zum Erstellen und Anzeigen von Annotationen innerhalb der Spielwelt

Der LBA-Dienst ist ein einfacher asynchroner Dienst, der die Kollaboration der Benutzer unterstützt und darüber hinaus eine sehr praktische Funktionalität bietet, die flexibel eingesetzt werden kann. Neben der Verwendung für die Dokumentation der virtuellen Welt, können auch andere Anwendungsszenarien oder Events realisiert werden wie beispielsweise eine Schnitzeljagd durch die virtuelle Umgebung.

7 Evaluation

Um die Umsetzbarkeit des Konzepts der virtuell kontextbasierten Dienste und der Middleware-Architektur zu überprüfen, wurde die VCBS-Middleware implementiert, zwei MMOGs wurden angebunden und Beispieldienste wurden entwickelt. Die Beschreibung der Implementierung, der angebundenen Spiele und der Beispieldienste befindet sich im vorangegangen Kapitel 6.

Im Folgenden werden zunächst Aufbau und Ziele der Evaluation dargestellt und anschließend die zur Evaluation durchgeführten Tests und Messungen, sowie die daraus resultierenden Ergebnisse vorgestellt. Dafür werden im zweiten Abschnitt die durchgeführten Tests zur Evaluation der entwickelten Funktionalität, inklusive der Anbindung an relevante Spiele und unter Verwendung von realistischen Diensten dargestellt. Im dritten Abschnitt werden dann die durchgeführten Messungen, unter Berücksichtigung großer Nutzerzahlen, verschiedener VCBS-Server und unterschiedlicher Diensttypen und deren Ergebnisse beschrieben.

7.1 Evaluationsaufbau und -ziele

Die Evaluation der in der Arbeit entwickelten Konzepte erfolgt anhand ihrer Umsetzung, d.h. mittels einer Evaluation der implementierten VCBS-Middleware. Verschiedene Evaluationsziele sind grundsätzlich bei einer Evaluation von Softwarekonzepten und deren Implementierung zu beachten. Zielsetzung im Rahmen dieser Dissertation ist zunächst ein Nachweis der allgemeinen Umsetzbarkeit des Konzeptes und der Realisierbarkeit der funktionalen Anforderungen mittels des implementierten Prototyps. Daneben sind auch die nicht-funktionalen Aspekte zu prüfen, zum einen die zuvor in Kapitel 5.1.2 vorgestellten spezifischen Kriterien, zum anderen die für eine Beurteilung von Softwareprodukten relevanten Merkmale.

Die ISO Norm [ISOb] zur Sicherstellung von Softwarequalität definiert neben der Funktionalität die folgenden Qualitätsmerkmale für Software:

- Zuverlässigkeit,
- Benutzbarkeit oder Benutzerfreundlichkeit,
- Effizienz,
- Wartbarkeit (Änderbarkeit) und
- Übertragbarkeit.

Diese Merkmale stimmen mit den nach dem etablierten FURPS Modell (*Functionality Usability Reliability Performance Supportability*) [Gra92] zu überprüfenden Merkmalen überein, wobei die Übertragbarkeit nicht Element des FURPS Modells ist.

Zur Überprüfung der unterschiedlichen Merkmale sind wiederum unterschiedliche Methoden oder Formen der Evaluation geeignet: **Funktionale Eigenschaften** sowie die **Zuverlässigkeit** (*Reliability*) lassen sich anhand der Implementierung mit Tests in verschiedenen Szenarien überprüfen. Der Nachweis der **Benutzerfreundlichkeit** (*Usability*) erfolgt üblicherweise mittels Benutzerstudien unter Einsatz verschiedenster Verfahren [Hol05]. Diese beinhalten Inspektionsmethoden, wie die heuristische Evaluation und Testmethoden, wie lautes Denken oder den Einsatz von Fragebögen. Dabei werden wiederum verschiedene Ziele adressiert [ISOa], nämlich Effektivität zur Lösung einer Aufgabe, Effizienz der Handhabung des Systems und Zufriedenheit der Nutzer einer Software. Die **Effizienz** (*Efficiency*) eines

Softwareproduktes beinhaltet die Antwort- und Verarbeitungszeiten eines Softwaresystems sowie den Durchsatz bei der Ausführung der Funktionen. Sie lässt sich durch Messungen des Systemverhaltens in realen Anwendungen überprüfen. Da dies in der Phase der Konzeption und Entwicklung oftmals nicht möglich ist, werden zur Evaluation der Effizienz oftmals Laborstudien oder Simulationen durchgeführt. Die **Wartbarkeit** (*Maintainability*) ist ein im Softwareengineering verwendetes Kriterium zur Beurteilung der Anpassbarkeit von Softwaresystemen. Dabei gibt es verschiedene Modelle zur Bestimmung des Wartbarkeitsindexes (*Maintainability Index*) [Ald01]. Die **Übertragbarkeit** (*Portability*) bezeichnet ein Merkmal, das im Softwareengineering von Bedeutung ist. Sie ist unmittelbar mittels der tatsächlichen Übertragung auf andere Rechnersysteme überprüfbar.

Die Arbeit schafft die Grundlagen für die Integration von Diensten in virtuelle Welten und damit eine Steigerung der Benutzerfreundlichkeit, d.h. insbesondere der Immersion. Weitere Usabilty-Aspekte hängen jedoch von den einzelnen Diensten ab, da deren Benutzerfreundlichkeit abhängig von deren Realisierung durch den Dienstersteller ist und somit nur auf Basis einzelner Dienste evaluiert werden kann.

Da der Schwerpunkt dieser wissenschaftlichen Arbeit auf der Entwicklung eines neuartigen Konzeptes liegt, beschränkt sich die Evaluation im Weiteren auf die funktionalen Eigenschaften und die Evaluation von ausgewählten Effizienzkriterien.

Die umfassendste Methode zur Evaluation von **Funktionalität** und **Effizienz** bestünde in dem realen Einsatz der Middleware in Kombination mit der Durchführung von Tests, einer Benutzerstudie und Messungen.

Dazu währe eine enge Zusammenarbeit mit Anbietern virtueller Welten notwendig um eine große Benutzerbasis anzusprechen, die eine Verwendung der VCBS-Middleware in vollem Umfang durchführen kann. Eine solche Evaluation bedarf jedoch der Kooperation mit mindestens einem Anbieter einer virtuellen Welt, da zuerst rechtliche Absprachen nötig sind, um nicht gegen die von den Anbietern verwendeten EULAs zu verstoßen. Die EULAs von MOGs dienen als Schutz vor Cheating und sind deshalb sehr restriktiv formuliert. Da die Spielanbieter jedoch sehr zurückhalten sind, um ungewollte Manipulationen an ihren Spielen zu vermeiden, ist es sehr schwierig, sie zu der Nutzung eines neuartigen Systems zu bewegen, wie es im Rahmen der Arbeit entwickelt wurde. Vor allem die Kooperation mit Anbietern von MOGs, die sehr große Benutzerzahlen aufweisen und daher für eine entsprechende Evaluation am besten geeignet wären, ist im Rahmen einer wissenschaftlichen Arbeit unrealistisch, da vor allem diese Anbieter in der Regel nur selbst entwickelte Komponenten für ihre Spiele einsetzen. Eine Alternative bestünde in der Entwicklung eines eigenen MOGs zum Zweck der Evaluation. Das würde jedoch bedeuten, dass zusätzlich noch eine Benutzerbasis für dieses MOG aufgebaut werden müsste, damit ein reales Szenario geschaffen werden kann. Dies würde jedoch den Rahmen der Arbeit sprengen.

Da die Evaluation in einem realen Szenario somit leider nicht möglich ist, wurden im Rahmen der Arbeit die Umsetzbarkeit des Konzepts und die funktionalen Eigenschaften anhand der prototypischen Implementierung mittels der Realisierung verschiedener Szenarien überprüft.

Zum Nachweis der Funktionalität von Konzept und Implementierung muss gezeigt werden, dass der Informationsaustausch zwischen einer existierenden virtuellen Welt und einem externen Dienst, der zusätzliche Funktionen oder Informationen bereitstellt, möglich ist und erfolgreich umgesetzt wurde. Zum zweiten muss gezeigt werden, dass das Konzept der virtuell kontextbasierten Dienste in virtuellen Welten wie vorgesehen möglich ist und erfolgreich umgesetzt wurde. Dies beinhaltet den Nachweis dass ein externer Dienst erfolgreich in die virtuelle Welt integriert wurde, dort in der spezifizierten virtuellen Situation dem Benutzer angeboten wird und bei der Nutzung situationsbezogen seine Informationen oder Funktionen bereitstellt.

Der Nachweis der Effizienz erfolgt mittels der Durchführung von Messungen in einer Laborsimulation unter Verwendung des implementierten Prototypen. Dabei erfolgt eine Beschränkung auf die Messung und Bewertung der Latenz, da diese für die Akzeptanz und Nutzbarkeit der Architektur das wichtigste Kriterium darstellt. Die Unterstützung einer großen Anzahl von Benutzern, d.h. die Skalierbarkeit der Architektur und die Nutzung verschiedenartiger Dienste werden betrachtet.

Der Nachweis der Funktionalität wird im folgenden Kapitel dargestellt. Danach folgt die Darstellung der Messungen, die zur Auswertung des Einflusses einer großen Anzahl von Benutzern sowie zur Bestimmung des Einflusses der Verwendung von Diensten auf die VCBS-Middleware durchgeführt wurden.

7.2 Evaluation der Funktionalität

Die VCBS-Middleware verfolgt zwei Hauptziele. Erstens die Verknüpfung von virtuellen Welten mit anderen Internet-Applikationen, sodass Informationen aus der virtuellen Welt dem Umfeld bereitgestellt und Dienste aus dem Umfeld in die virtuelle Welt zu integrieren werden können. Zweitens die Realisierung von kontextbasierten Diensten in virtuellen Umgebungen, sodass Dienste innerhalb der virtuellen Welt, passend zu der virtuellen Situation des Benutzers (kontextbasiert), zur Verfügung gestellt werden können (siehe Kapitel 2.2). Die formulierten Ziele wurden erreicht und die Implementierung der VCBS-Middleware stellt die benötigten Funktionen zur Verfügung.

Zur Überprüfung der Funktionalität, die die VCBS-Middleware bietet, wurden zwei realistische Testszenarien entwickelt. Das erste Szenario dient zur Validierung der ersten primären Zielsetzung, der Realisierung des Informationsaustauschs. Mittels der VCBS-Clients wird *World of Warcraft*, als aktuell wichtigster Vertreter der MMORPGs, angebunden und als Beispieldienst der LBA-Dienst eingesetzt. Im zweiten Szenario wurde als beispielhafter VCBS-Dienst "Fisherman's Friend" in *Ultima Online* integriert, da dieses sehr flexible Verwendungsmöglichkeiten bietet. Es dient zur Validierung der zweiten primären Zielsetzung der Realisierbarkeit von kontextbasierten Diensten.

7.2.1 Informationsaustausch

Das erste Hauptziel der VCBS-Middleware ist es, einen Informationsaustausch zwischen einer virtuellen Welt und externen Internet-Applikationen zu ermöglichen. Der von der VCBS-Middleware bereitgestellte Informationsaustausch zwischen einem Spiel-Client und einer Internet-Applikation, die einen Dienst bereitstellt, umfasst vier Schritte. Abbildung 7.1 visualisiert die für den Informationsaustausch nötigen Informationsflüsse.

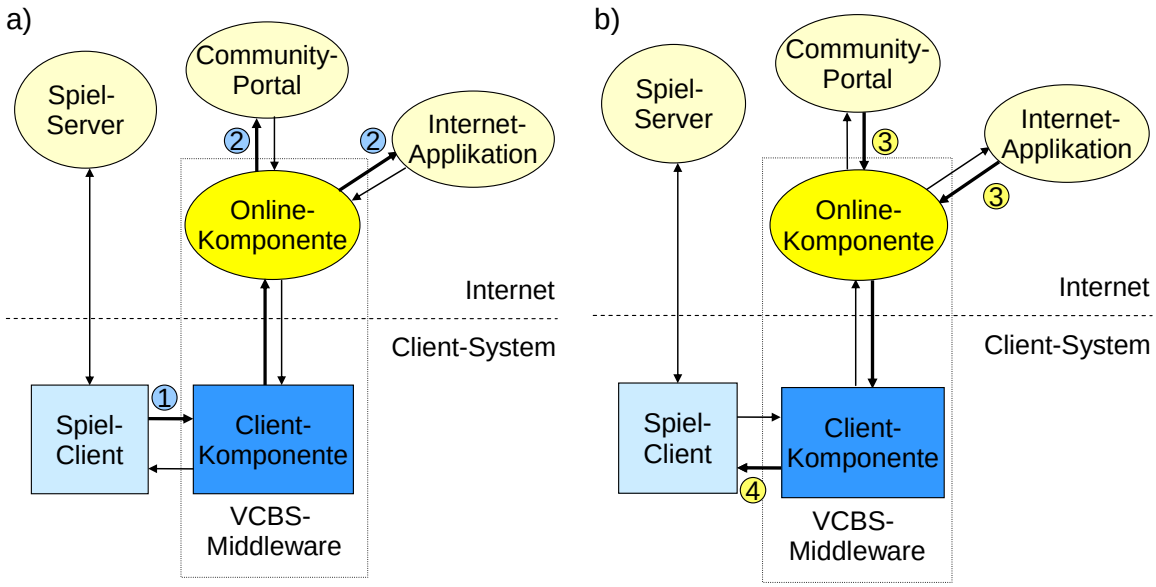

Abbildung 7.1: Informationsaustausch zwischen Spiel und Internet-Applikationen: a) Informationsfluss aus dem Spiel heraus b) Informationsfluss in das Spiel hinein

- Informationsfluss aus dem Spiel heraus (a):
  1. Erfassung von in-game Informationen (Kontextinformationen) aus der virtuellen Welt
  2. Verteilung der in-game Informationen an die aktiven Dienste
- Informationsfluss in das Spiel hinein (b):
  3. Übertragung der vom Dienst bereitgestellten Informationen (Servicedaten)
  4. Integration der vom Dienst bereitgestellten Informationen in die virtuelle Umgebung

Zur Validierung der mit der VCBS-Middleware realisierten Informationsflüsse wird ein Informationsaustausch zwischen *World of Warcraft* und dem Community-Portal *gamerlist.net* umgesetzt. *gamerlist.net* stellt einen Dienst zur lokationsbasierten Annotation (LBA-Dienst, siehe Kapitel 6.3.4)) von virtuellen Umgebungen bereit.

1) Erfassung von in-game Informationen (Kontextinformationen) aus der virtuellen Welt

Für die Verbindung zwischen dem VCBS-Client und dem Spiel-Client von *World of Warcraft* wird ein entsprechendes Addon (siehe auch 6.2.2) eingesetzt. Wenn der Spieler den LBA-Dienst in *World of Warcraft* verwendet, dann wird ein neues Fenster mit dem Titel "*Marker Manager*" angezeigt (siehe auch Abbildung 6.8). Dieses Fenster kann frei positioniert werden. Der Marker Manager enthält eine Schaltfläche zum Hinzufügen von Annotation und ermöglicht dem Spieler, eine Situation oder ein Objekt im Spiel zu beschreiben, indem er seine aktuelle Position mit einer Notiz versieht. Dazu öffnet sich ein Dialog ("*Add Marker*") im Spielinterface mit welchem neue Annotation erfasst werden können. Der Dialog Add Marker verfügt über zwei Eingabefelder, "*Name*" und "*Description*" (siehe Abbildung 7.2). Ein Klick auf "*Save*" speichert die Annotation im Addon ab.

Abbildung 7.2: Verwendung des "Add Marker"-Dialoges zur Erstellung von Annotationen innerhalb des Spiels

Die Kontextinformationen des virtuellen Charakters, insbesondere seine Lokation, werden über das Addon automatisiert erfasst. Der VCBS-Client kann die im Addon erfassten Informationen (Kontextinformationen, wie die Lokation, und Annotationen des Spielers) an einen VCBS-Server weiterleiten.

2) Verteilung der in-game Informationen an die aktiven Dienste

Nachdem der VCBS-Server die Informationen erhalten hat, leitet er die relevanten Kontextinformationen an die vom Benutzer aktivierten Dienste, im realisierten Beispiel an den LBA-Dienst, weiter. Dabei

werden an jeden Dienst nur die jeweils benötigten Kontextinformationen (wie in diesem Beispiel die Lokation des virtuellen Charakters) weitergegeben. Zusätzlich werden die Benutzereingaben für den Dienst (die Annotation des Benutzers) an den LBA-Dienst weitergegeben. Der LBA-Dienst speichert die Annotation des Benutzers zusammen mit den Positionsangaben. Im Evaluationsszenario wird der LBA-Dienst über das Community-Portal *gamerlist.net* bereitgestellt. *gamerlist.net* stellt die Annotation des Benutzers zum einen auf der Webseite des Community-Portals dar und stellt sie zusätzlich direkt den Freunden (*Buddies*) des Benutzers zur Verfügung. Das Portal erlaubt darüber hinaus das spätere Editieren der Annotation durch den Benutzer, der sie im Spiel erstellt hat, so dass zum Beispiel die im Spiel erstellten Annotationen später überarbeitet und erweitert werden können (siehe Abbildung 7.3).

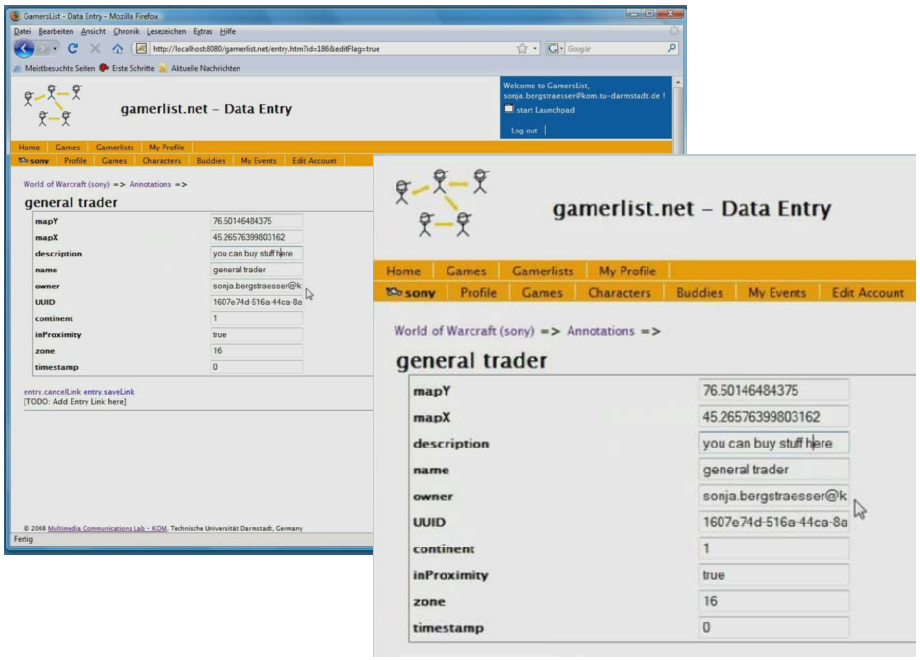

Abbildung 7.3: Darstellung und Überarbeitung der Annotation auf dem Community-Portal *gamerlist.net*

3) Übertragung der vom Dienst bereitgestellten Informationen (Servicedaten)

Die Annotation kann nun an den Benutzer und seine Freunde verteilt werden. Dafür stellt das Community-Portal *gamerlist.net* mit dem LBS-Dienst die Annotation als Servicedaten bereit. Über einen VCBS-Server werden sie an den VCBS-Client der Dienstbenutzer übertragen.

4) Integration der vom Dienst bereitgestellten Informationen in die virtuelle Umgebung

Der VCBS-Client empfängt die Servicedaten, d.h. die Annotationen und deren Lokationsinformationen und macht sie über das Addon im Spiel zugänglich. Wenn nun der Benutzer, oder einer seiner Freunde, das nächste Mal zu der virtuellen Lokation einer Annotation kommt, dann öffnet sich innerhalb des Spielinterfaces ein kleines Fenster ("*Marker Touched*"), das die entsprechende Annotation zu dieser Position anzeigt (siehe Abbildung 7.4).

Neben der Möglichkeit, neue Annotationen zu erstellen, werden im Marker Manager die bereits erstellten Annotationen mit Namen gelistet, alte werden blau, neue grün und erreichte rot angezeigt (siehe auch Abbildung 6.8). Zusätzlich enthält das Fenster noch Zusatzinformationen zur aktuellen Position des virtuellen Charakters des Benutzers wie die Koordinaten der aktuellen Position oder das Gebiet, in dem sich der virtuelle Charakter aktuell befindet.

Abbildung 7.4: Anzeige einer Annotation innerhalb des Spiels über das "Marker Touched"-Fenster

7.2.2 Kontextbasierte Bereitstellung von Diensten in virtuellen Welten

Das zweite Haupziel der VCBS-Middleware ist es, die Bereitstellung von Diensten in virtuellen Welten basierend auf dem virtuellen Kontext des Benutzers zu realisieren. Die VCBS-Middleware setzt dabei das Prinzip der Dienstauswahl durch den Benutzer um (vergleiche Kapitel 3.1.3). Das bedeutet, die Ausführung des Dienstes wird manuell vom Benutzer aktiviert, die bereitgestellten Informationen werden jedoch automatisch an die virtuelle Situation angepasst. Die von der VCBS-Middleware umgesetzte Integration und Bereistellung entsprechender Dienste in einer virtuellen Umgebung umfasst drei Funktionalitäten:

1. **Dienstangebot**: Darstellung des kontextbasierten Dienstangebots in einer virtuellen Umgebung mit der Möglichkeit der Aktivierung und Deaktivierung von Diensten.
2. **Dienstnutzung**: Benutzerinteraktion mit den Diensten und Darstellung der Informationen, die ein Dienst bereitstellt, innerhalb der virtuellen Umgebung.
3. **Dienstanpassung**: Automatische Updates der von den verwendeten Diensten bereitgestellten Informationen und kontextbasierte Deaktivierung von Diensten.

Zur Validierung der mit der VCBS-Middleware realisierten Umsetzung der kontextbasierten Bereitstellung von Diensten in virtuellen Welten wird im Spiel *Ultima Online* der von einer Web-Applikation erbrachte "Fisherman's Friend" Dienst (siehe auch 6.3.3) bereitgestellt.

1) Dienstangebot

Der "Fisherman's Friend" Dienst dient zur Unterstützung des Angelns in virtuellen Umgebungen. Aus diesem Grund ist er nur in bestimmten virtuellen Situation, wenn der Spieler Angeln gehen möchte und sich in der Nähe eines virtuellen Gewässers aufhält, für den Benutzer nützlich. In anderen Situationen ist dieser Dienst überflüssig. Die Anzeige unnötiger Zusatzinformationen kann für den Benutzer störend sein. Folgende Vorbedingungen (*Precondition Parameter*) müssen erfüllt sein damit der "Fisherman's Friend" Dienst gültig ist und dem Benutzer angeboten wird: der virtuelle Charakter des Benutzer muss sich

an einer Angelposition (in der Nähe eines virtuellen Gewässers) aufhalten und einen zum Angeln geeigneten Gegenstand (zum Beispiel eine Angelrute) in der Hand halten. Treffen diese Vorbedingungen zu, dann wird der "Fisherman's Friend" Dienst dem Benutzer angeboten (siehe Abbildung 7.5).

Abbildung 7.5: Anzeige der verfügbaren Dienste im VCBS Service Browser

Das Angebot der Dienste innerhalb der virtuellen Welt wird über ein Service User Interface realisiert (siehe auch 5.3.3). In der beispielhaften Implementierung wird das Service User Interface als "*VCBS Service Browser*" im Spiel dargestellt. Der Service Browser listet alle aktuell verfügbaren Dienste auf und bietet eine Schaltfläche zum aktivieren von Diensten an. Neben der Aktivierung kann sich der Benutzer über den Service Browser weitere Angaben zu einem Dienst, wie eine Dienstbeschreibung oder die vom Dienst verwendeten Kontextinformationen, anzeigen lassen.

2) Dienstnutzung

Hat der Benutzer den "Fisherman's Friend" Dienst aktiviert, dann öffnet sich ein kleines Fenster "*Fishing Statistics*". Diese zeigt die verschiedenen Fischarten, die an der Angelposition des Charakters gefangen werden können und die Wahrscheinlichkeit (*Probabilities*), mit der die Art gefangen wird an (siehe Abbildung 7.6). Während ein Benutzer den Dienst verwendet, werden dem externen "Fisherman's Friend" Dienst die Positionskoordinaten des virtuellen Charakters und die Resultate seiner Angelaktionen von der VCBS-Middleware übermittelt. Aus den von allen Benutzern, die den Dienst verwenden, erhaltenen Daten berechnet der "Fisherman's Friend" Dienst dann die Wahrscheinlichkeiten für die entsprechenden Fischarten an verschiedenen Angelpositionen. Die ermittelten Statistiken werden wiederum allen Benutzern des Dienstes bereitgestellt.

3) Dienstanpassung

Nach jedem Fang eines beliebigen Dienstnutzers erhält der Dienst von der VCBS-Middleware neue Daten, woraufhin der Dienst seine Statistiken aktualisiert. Die aktualisierten Statistiken werden wieder an den Benutzer übertragen. Da an verschiedenen Angelpositionen unterschiedliche Fische mit unterschiedlichen Wahrscheinlichkeiten gefangen werden können, werden die vom Dienst bereitgestellten Statistiken auch angepasst, sobald sich die Angelposition des virtuellen Charakters ändert. Abbildung 7.7 zeigt den virtuellen Charakter der vorher auf dem Steg geangelt hatte und nun auf die Landzunge gegangen ist. Für seine neue Position bekommt er automatisch die aktuelle Angelstatistik (die an dieser

Abbildung 7.6: Nutzung des "Fisherman's Friend" Dienstes: Anzeige der Fischfang-Statistik

Stelle andere Werte und einen zusätzlichen Fisch beinhaltet) vom "Fisherman's Friend" Dienst bereitgestellt.

Abbildung 7.7: Anzeige der Fischfang-Statistik für die geänderte Angelposition

7.3 Evaluation der Effizienz

Die Effizienz-Eigenschaften, die in der durchgeführten Evaluation untersucht wurden, sind die zeitnahe Dienstbereistellung, die Unterstützung einer großen Anzahl an Benutzern (Skalierbarkeit) und die Unterstützung verschiedenartiger Dienste. Im folgenden Teil der Evaluation wird gezeigt, dass die VCBS-Middleware auch in einem realistischen Szenario mit vielen Benutzern eingesetzt werden kann.

Für die Bewertung der Eigenschaften werden die entstehenden Latenzen der Kommunikation zwischen dem VCBS-Client und dem VCBS-Server betrachtet. Die Latenz stellt für die Akzeptanz und Nutzbarkeit der Architektur das wichtigste Kriterium dar und hat eine zentrale Bedeutung im Bereich der MOGs (sie-

he Kapitel 3.2.3). Auch für Zusatzdienste, wie die durch die VCBS-Middleware bereitgestellten Dienste, ist die Latenz relevant, da der Benutzer auch für die Interaktion mit einem in die virtuelle Welt integrierten Dienst entsprechende Reaktionszeiten erwartet. Dabei kann die Middlware nur die Kommunikationszeiten und die Verarbeitungszeiten der Middleware-Komponenten beeinflussen. Die vom Dienst benötigten Verarbeitungszeiten sind abhängig von dem jeweiligen externen Dienst. Für unterstützende Dienste sind jedoch höhere Latenzen akzeptabel als für die MOGs selbst, da diese sich nicht direkt auf die Spielsteuerung auswirken. Eine geringe Latenz kann auch hier das empfundene Spielerlebnis unterstützen, beispielsweise durch das zeitnahe Angebot neuer Dienste, und ist somit erstrebenswert.

Die in den zur Evaluation durchgeführten Messreihen betrachtet Latenz beinhaltet die Kommunikationszeit zwischen VCBS-Client und VCBS-Server sowie die Verarbeitungszeit auf dem VCBS-Server. Die Kommunikation beinhaltet dabei die Nachrichtenübertragung vom VCBS-Client an den VCBS-Server und die Übertragung der Antwortnachricht vom VCBS-Server an den VCBS-Client. Die Dauer der Kommunikation mit den externen Diensten wird nicht betrachtet, da dieser Teil der Kommunikation durch den VCBS-Server entkoppelt ist. Das bedeutet, dass die Antwortnachrichten vom VCBS-Server an den VCBS-Client unabhängig von der Kommunikation zwischen VCBS-Server und den externen Diensten ist. Das für die Client-Server Kommunikation verwendete Kommunikationsprotokoll wurde in Kapitel 6.1.3 vorgestellt (siehe Abbildung 6.4). Die Kommunikation wurde in der prototypischen Implementierung der VCBS-Middleware durch die Verwendung von Web Services realisiert. Die Annahme dabei ist, dass die verwendete Kommunikationstechnik die prinzipiellen Anforderungen erfüllen kann. Die entstehenden Latenzen also nicht zu groß werden (die Latenzen sollten 2000ms nicht überschreiten). Eine weitere Annahme ist, dass, durch die von der VCBS-Middleware durchgeführte Verteilung der Benutzer auf verschiedene VCBS-Server, die Middleware skalierbar ist und somit auch für große Benutzerzahlen eingesetzt werden kann. Bezüglich der Verwendung verschiedenartiger Dienste, wird ein Einfluss der Dienstart auf die Latenz erwartet, die Annahme dabei ist, dass sich dieser Einfluss jedoch nicht im großen Maße auf die Latenz auswirkt.

Zur Messung der entstehenden Latenzen wurde ein möglichst realistisches Szenario nachgebildet. Dazu wurde die zuvor vorgestellte Implementierung der VCBS-Middleware auf verschiedenen Rechnern des G-Labs instanziiert. G-Lab (german-lab [1]) ist eine Studien- und Experimentalplattform für das Internet der Zukunft. Es ermöglicht Tests in einer realitätsgerechten Umgebung. Die verschiedenen G-Lab Standorte bestehen jeweils aus bis zu 25 Knoten und sind über ganz Deutschland verteilt. Neben dem Standort in Darmstadt gibt es unter anderem Knoten in Würzburg, Kaiserslautern, München und Berlin. Die verschiedenen Teile der Online-Komponente der VCBS-Middleware, die VCBS-Registry und verschiedene VCBS-Server, wurden dabei auf verschiedene Knoten in Darmstadt verteilt. Das Verhalten der Benutzer wurde simuliert. Dazu wurde JMeter [4] eine Java-Applikation für funktionale Tests und Performance-Messungen verwendet und es wurden JMeter-Instanzen auf verschiedene entfernte G-Lab Knoten verteilt. Für das Messen aus der Perspektive eines Clients, ist JMeter bezüglich SOAP sehr gut geeignet. Jede JMeter-Instanz simuliert eine variable Menge von VCBS-Clients. Als Dienste wurden keine realen VCBS-Dienste verwendet, sondern künstliche Dienste, die spezielle Eigenschaften aufweisen und miteinander verglichen werden können. Diese Dienste wurden ebenfalls auf andere entfernte G-Lab Knoten verteilt.

Die Latenz der Client-Server Kommunikation wird beeinflusst von verschiedenen Faktoren. Dazu zählen verschiedene Parameter zur Variation des Kommunikationsverhaltens, die Nutzeranzahl, das Nutzerverhalten und die Art der verwendeten Dienste. Um den Einfluss dieser Faktoren auf die Latenz zu messen, wurden sechs verschiedene Messreihen mit jeweils geänderten Faktoren erfasst und bewertet. In den ersten beiden Messreihen werden grundlegende Kommunikationsparameter der VCBS-Middleware variiert. In den nächsten beiden Messreihen variiert die Benutzeranzahl. Sie gibt Aufschluss über die Skalierbarkeit der Implementierung der Architektur. Die letzten beiden Messreihen variieren Verwendung und Eigenschaften der Dienste, die dem Benutzer über die VCBS-Middleware zur Verfügung gestellt werden.

Die Evaluation der vorgestellten Effizienz-Eigenschaften deckt durch die Fokussierung auf die Latenz und die Untersuchung der drei vorgestellten Einflussfaktoren die wichtigsten Aspekte für die VCBS-Middleware ab. Weitere Aspekte zu betrachten, würde den Rahmen der Arbeit sprengen.

Simulation des Benutzerverhaltens und verwendete Dienste

Bei der Durchführung der Messungen wird ein künstlicher Echo-Service verwendet. Der Echo-Service sendet immer alle Daten, die er von der VCBS-Middleware erhält, also alle virtuellen Parameter und alle vom Benutzer generierten Daten, als Servicedaten wieder zurück. Er erwartet einen kompletten Satz virtueller Parameter (*Required Parameter* umfassen die komplette xgdl) und ist in jeder virtuellen Situation gültig (hat keine *Precondition Parameter*). Er erzeugt bei Verwendung einen stetigen Informationsfluss in beide Richtungen, wodurch ein stetiger Informationsaustausch zwischen Benutzer und Dienst entsteht. Der Echo-Service stellt dadurch einen worst case Dienst dar, da er den Informationsaustausch maximal nutzt.

Zur Durchführung der Messreihen muss, wie zuvor beschrieben, das Benutzerverhalten simuliert werden. Die Simulation des Benutzerverhaltens über JMeter umfasst alle Aktionen, die ein Benutzer der VCBS-Middleware ausführen kann. Dazu werden die entsprechenden Nachrichten verwendet, die der Web Service des VCBS-Clients erzeugt (siehe Standardablauf der Kommunikation für einen Benutzer in Abbildung 7.8).

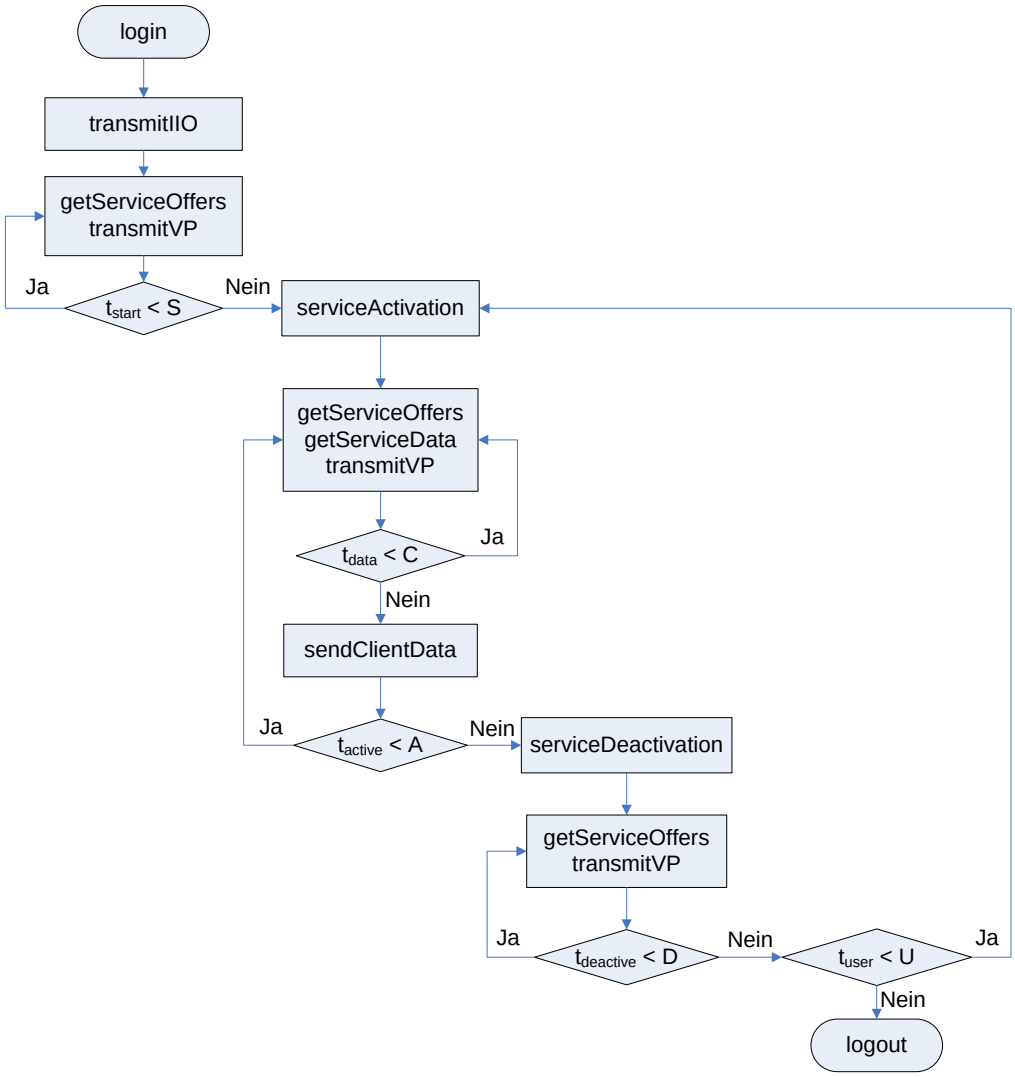

Abbildung 7.8: Standardablauf der Kommunikation für einen Benutzer

Zuerst erfolgt die Anmeldung (*login*) bei der VCBS-Middleware, worauf der normale Kommunikationsablauf zwischen VCBS-Client und VCBS-Server beginnt (siehe auch Kapitel 6.1.3). Zunächst überträgt jeder Benutzer einen kompletten Satz der virtuellen Parameter (*transmitIIO*). Dann beginnt die *Start-Phase S*. Sie beinhaltet die Anfrage nach verfügbaren Diensten (*getServiceOffers*). Diese Anfrage führt jeder VCBS-Client jede Sekunde durch. Als Reaktion wird der Echo-Service jedem Benutzer angeboten, da dieser ja in jeder virtuellen Situation gültig ist. Für die Simulation wird angenommen, dass die Aktivitäten, die ein Benutzer durchführt, zu einer Änderung von fünf virtuellen Parametern jede Sekunde führen. Diese Änderung triggert jeweils eine Übertragung der virtuellen Parameter (*transmitVP*).

Nach Ende der Start-Phase aktiviert der Benutzer den angebotenen Dienst (*serviceActivation*). Die *Aktiv-Phase A* besteht aus zwei Teilen. Während der *Kontext-Phase C* wird die Übertragung von virtuellen Kontextinformation durch virtuelle Parameter durchgeführt, die verfügbaren Dienste werden abgefragt und zusätzlich werden die Daten des aktiven Dienstes von der Middleware abgerufen (*getServiceData*). Nach dem Ablauf einer Kontext-Phase werden dann einmal zusätzliche Daten für den Dienst an die Middleware übertragen (*sendClientData*). Darauf folgt dann die nächste Kontext-Phase. Die Dienstnutzung wird nach Ablauf der Aktiv-Phase durch den Benutzer beendet (*serviceDeactivation*). In der nun folgenden *Passiv-Phase D* wird kein Dienst verwendet. Aktive und passive Phase wechseln sich solange ab, bis die komplette Nutzungsdauer *Nutzungs-Phase U* des Benutzers abgelaufen ist und dieser sich abmeldet (*logout*).

Nicht in allen nachfolgend beschriebenen Messreihen wurden die Simulation des Benutzerverhaltens in der hier vorgestellten Form und der vollständige Echo-Service genutzt. Variationen werden explizit angegeben. Die Bestimmung der Latenz erfolgte jeweils durch mindestens zwei valide Messungen, die in allen Fällen keine großen Varianzen zueinander aufwiesen. Die durchgeführten Messungen waren nahezu reproduzierbar.

7.3.1 Einfluss von Parametern zur Steuerung des Kommunikationsverhaltens

Das Kommunikationsverhalten der VCBS-Middleware lässt sich über verschiedene Parameter steuern. Dazu zählen insbesondere die Häufigkeit der Übertragung der virtuellen Kontextinformationen und die Abfragerate nach neuen Dienstangeboten. Der Einfluss dieser Parameter auf die Latenz wurde in den ersten beiden Messreihen untersucht. Die Kommunikation (Anzahl und Größe der übertragenen Nachrichten) zwischen dem VCBS-Client und dem VCBS-Server (siehe Kapitel 6.1.3) wird vor allem durch die Häufigkeit der Übertragung der virtuellen Parameter (*transmitVP* bzw. *transmittIIO*), sowie der, für die Abfrage der Dienstangebote (*getServiceOffers*) gewählten Abfragerate bestimmt (vergleiche Abbildung 6.4).

Die **Häufigkeit der Übertragung der virtuellen Parameter** hängt von den Aktionen des Spielers in der virtuellen Welt und den sich daraus ergebenden Änderungen der virtuellen Parameter ab. Ein sehr stark variierendes Benutzerverhalten ist typisch für MMORPGs. Das belegen auch Untersuchungen ([SDM09]), die zeigen, dass Spieler ganz unterschiedliche Phasen bezüglich ihrer Aktivität haben. So gibt es Spielsituationen, in denen ein Spieler eher passiv ist und über einen längeren Zeitraum kaum Änderungen der virtuellen Parameter stattfinden, und andere Spielsituationen, in denen ein Spieler hoch aktiv ist und sich dadurch auch seine virtuelle Situation sehr schnell ändern kann. Der *Parameter-Cache* im VCBS-Client (siehe Kapitel 6.1.1) ermöglicht jedoch das Sammeln der geänderten Parameterwerte. Somit kann ein minimaler Zeitabstand zwischen zwei Übertragungen virtueller Parameter definiert werden (*Updaterate*). Diese künstliche Verzögerung kann das entstehende Kommunikationsaufkommen stark reduzieren. Der minimale Zeitabstand definiert den worst case, der einer maximalen Änderungshäufigkeit der virtuellen Situation im Spiel (hohe Spieleraktivität) entspricht. Eine Variation des minimalen Zeitabstands beeinflusst das Kommunikationsaufkommen der Middleware und erhöht die Last auf dem VCBS-Server, da für jedes Update der virtuellen Parameter die aktuelle virtuelle Situation abgeleitet wird. Die Updaterate ist der erste wichtige Einflussfaktor auf die entstehenden Latenzen.

Die **Abfragerate der Dienstangebote** ist unabhängig von spielinternen Events. Die Abfrage der verfügbaren Dienstangebote wird im regelmäßigen Kommunikations-Loop der VCBS-Middleware ausgeführt (siehe Abbildung 6.4), um regelmäßig alle für die aktuelle virtuelle Situation des Benutzers passenden Dienste zu enthalten. Der Abstand zwischen zwei Abfragen kann variiert werden (*Abfragerate*). Eine hohe Abfragerate ermöglicht ein sehr zeitnahes Angebot neuer Dienste an den Benutzer. Je höher die Abfragerate, desto zeitnäher können dem Benutzer neu verfügbare Dienste angeboten werden, aber desto höher wird auch das Kommunikationsaufkommen zwischen Client und Server. Die Abfragerate ist der zweite wichtige Einflussfaktor auf die entstehenden Latenzen.

Bei der Durchführung der Messungen wurden die zwei Parameter Updaterate und Abfragerate variiert, um deren Einfluss auf die entstehende Latenz zu untersuchen. Der Simulationsablauf basiert auf dem zuvor beschriebenen Ablauf (siehe Abbildung 7.8). Die Laufzeit der Tests beträgt knapp 1,5 Stunden (5200 Sekunden). Es wurden 20 JMeter-Instanzen verwendet, die jeweils acht Benutzer simulieren. Das sind für die Messungen insgesamt 160 Benutzer. Die weiteren Parametern der Simulation des Benutzerverhaltens sind in Tabelle 7.1 aufgelistet.

Tabelle 7.1: Parameter: Updaterate und Abfragerate

Parameter	Zeitdauer (s)
Start-Phase S	30
Kontext-Phase C	60
Aktiv-Phase A	240
Passiv-Phase D	30
Nutzungs-Phase U	3600

Variation der Updaterate

Für die erste Messreihe wurden die Simulationen jeweils in zwei Varianten durchgeführt. In Variante A, wird jede Sekunde ein neues Update (Updaterate $u = 1$) der virtuellen Parameter gesendet (*transmitVP*), und in Variante B, werden jede Sekunde zwei $u = 2$ Updates gesendet. Die Übertragung des kompletten Parametersatzes (*transmitIIO*) wird in dieser Konfiguration nur einmal nach der Anmeldung des Benutzers durchgeführt und kann aus diesem Grund bei der Betrachtung zunächst vernachlässigt werden. Für die Abstände zwischen den Anfragen der Dienstangebote (*getServiceOffers*) wurden unterschiedliche Zeiten (Abfragerate a) verwendet.

Da, wie beschrieben, die Kommunikation primär durch die beiden Kommunikationsnachrichten *getServiceOffers* und *transmitVP* beeinflusst wird, werden sie zunächst einzeln betrachtet.

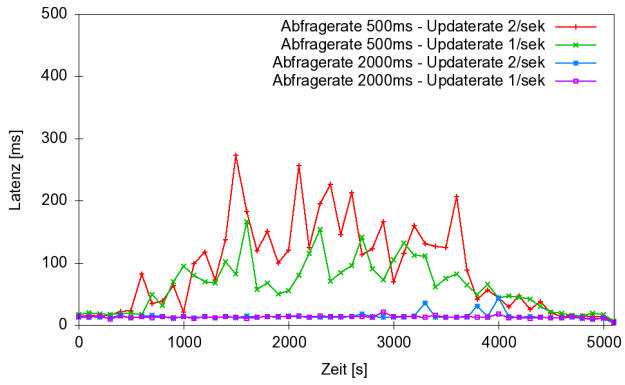

Abbildung 7.9: Latenzen für *getServiceOffers*

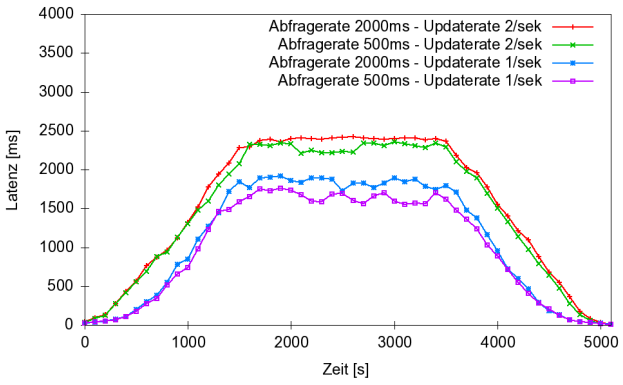

Abbildung 7.10: Latenzen für *transmitVP*

Abbildung 7.9 zeigt die Latenzen für *getServiceOffers* bei den Abfrageraten $a = 500ms$ und $a = 2000ms$ und den Updateraten $u = 1$ und $u = 2$. Die entstehende Latenz beträgt im Schnitt 75,9ms bei $a = 500ms$ und 12,9ms bei $a = 2000ms$. Die Variation der Updaterate führt bei einer niedrigen Abfragerate $a = 2000ms$ zu einem unwesentlichen Unterschied in der Latenz, er beträgt im Schnitt 8 % (1,2ms). Bei einer hohen Abfragerate $a = 500ms$ wächst der Unterschied, er beträgt im Schnitt 30 % (27,4ms). Einen größeren Einfluss hat die Abfragerate. Bei einer niedrigen Updaterate von 1/Sekunde ($u = 1$) variiert die Latenz um 80 % und bei einer hohen Updaterate von 2/Sekunde ($u = 2$) um 85 %.

Abbildung 7.10 zeigt die Latenzen für *transmittVP* bei den Abfrageraten $a = 500ms$ und $a = 2000ms$ und den Updateraten $u = 1$ und $u = 2$. In diesem Fall hat vor allem die Updaterate einen Einfluss auf die Latenz. Für eine niedrige Abfragerate $a = 2000ms$ liegt der Unterschied in der Latenz bei 29 % und bei einer hohen Abfragerate $a = 500ms$ bei 32 %. Die verwendete Abfragerate hat nur geringe Auswirkungen. Bei einer niedrigen Updaterate ($u = 1$) beträgt der Unterschied der Latenz im Durchschnitt 9,85 % und bei einer hohen Updaterate ($u = 2$) nur noch 4,69 %.

Die gemessene Latenz ist abhängig von der Benutzeranzahl. Die resultierenden Kurven steigen zu Beginn der Messungen an, da sich die Benutzer zunächst nacheinander anmelden und Fallen zum Ende hin wieder ab, da sich die Benutzer nach Ende ihrer Nutzungs-Phase wieder nacheinander abmelden.

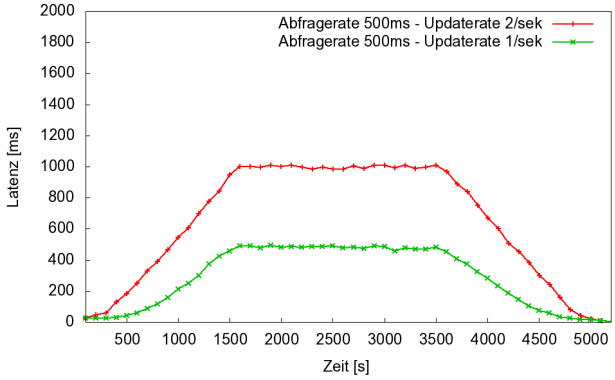

Abbildung 7.11: Latenzen der Kommunikation bei einer Abfragerate von 500ms

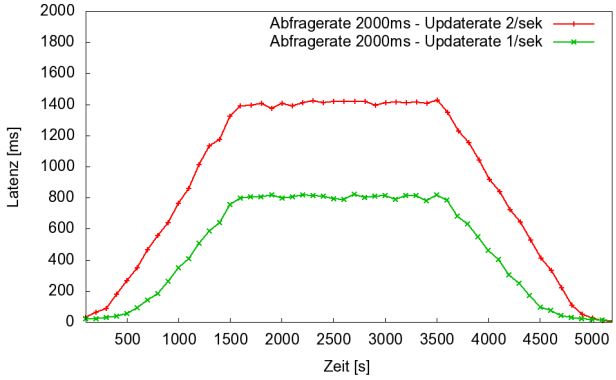

Abbildung 7.12: Latenzen der Kommunikation bei einer Abfragerate von 2000ms

Tabelle 7.2: Updaterate

Abfragerate a [ms] / Updates u pro Sekunde	∅ Latenz	∅ Zunahme
500 / 1	287,4	
500 / 2	637,5	54,9 %
2000 / 1	475,3	
2000 / 2	896,7	46,9 %
4000 / 1	528,2	
4000 / 2	956,1	44,7 %
Gesamte ∅ Zunahme		48,8 %

Die zwei betrachteten Parameter wirken sich, wie erwartet, jeweils unterschiedlich auf die beiden betrachteten Kommunikationsnachrichten (*getServiceOffers* und *transmittVP*) aus. Um dies und den Einfluss der anderen Nachrichtenübertragungen zwischen dem VCBS-Client und dem VCBS-Server zu berücksichtigen, werden im Folgenden die akkumulierten Ergebnisse aller Kommunikationsnachrichten (mit *transmittIIO*, *serviceActivation* und *sendClientData*, siehe auch 6.1.3) im Vergleich der beiden Varianten A

($u = 1$) und B ($u = 2$) dargestellt (siehe Abbildungen 7.11 für die Abfragerate $a = 500ms$ und 7.12 für $a = 2000ms$).

Die Ergebnisse zeigen, dass die verwendete Updaterate insgesamt für alle Nachrichten einen deutlichen Effekt auf die gemessenen Latenzen hat. Bei den akkumulierten Ergebnissen liegen die gemessenen durchschnittlichen Latenzzeiten zwischen 287,4ms (A 500ms) und 956,1 (B 4000ms). Tabelle 7.2 zeigt eine Zusammenfassung der Ergebnisse für die Abfrageraten $a = 500ms$, $a = 2000ms$ und $a = 4000ms$ für beide Varianten A und B. Durch eine Verdopplung der Parameterübertragung (Updaterate) von einer Übertragung pro Sekunde auf zwei Übertragungen pro Sekunde erhöht sich sowohl das Kommunikationsaufkommen zwischen VCBS-Client und VCBS-Server als auch die Verarbeitung der virtuellen Parameter auf dem VCBS-Server, was in einer Zunahme der durchschnittliche Latenz um fast 50 % (48,8 %) resultiert.

Variation der Abfragerate

In der zweiten Messreihe wurde die Updaterate auf $u = 1$, d.h. ein Update pro Sekunde festgelegt. In dieser Messreihe wurde der Einfluss der Abfragerate genauer betrachtet. Auch die Durchführung dieser Messreihe basiert auf der vorgestellten Simulation des Benutzerverhaltens mit den oben definierten Parametern (siehe Tabelle 7.1). Als Abfrageraten, also für die Zeit zwischen den Anfragen (*getServiceOffers*), wurden Zeiten von 200ms, 500ms, 1s, 2s und 4s verwendet. Abbildung 7.13 zeigt die durchschnittlichen Latenzzeiten für die betrachteten Abfrageraten.

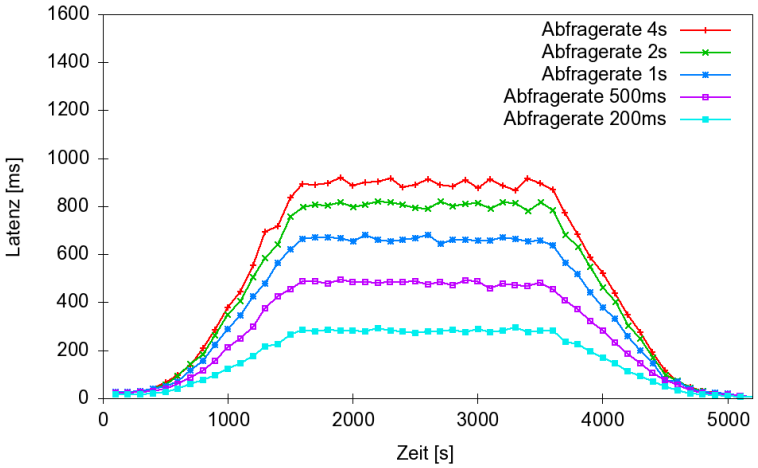

Abbildung 7.13: Latenzen der Kommunikation bei unterschiedlichen Abfrageraten

Tabelle 7.3: Abfragerate

Abfragerate a [ms]	∅ Latenz	Differenz
200	170	
500	287,4	40,8%
1000	386,05	25,5%
2000	475,3	18,7%
4000	528,2	10%

Wie bei der Betrachtung der Latenzen für die Übertragung der virtuellen Parameter (*transmitVP*, siehe Abbildung 7.10) resultiert eine niedrige Abfragerate in höheren Latenzen als eine höhere Abfragerate. Dabei wird der Unterschied kleiner, je niedriger die Abfragerate gewählt wird. Bei einer Rate von 200ms sind die Latenzen im Schnitt noch um 40,8% geringer als bei einer Rate von 500ms. Der Unterschied

zwischen 2s und 4s liegt nur noch bei 10%. Dieses Verhalten der VCBS-Middleware ist sehr günstig, da mit einer hohen Abfragerate auch eine hohe Aktualität der vorliegenden Dienstangebote erreicht werden kann. Tabelle 7.3 fasst die Messergebnisse zusammen.

7.3.2 Einfluss der Benutzeranzahl

Zur Untersuchung der Eigenschaften der VCBS-Middleware bezüglich variierender Benutzerzahlen wurden zwei weitere Messreihen durchgeführt. Dabei wurde die Latenz der Kommunikation zwischen den Benutzern und den betrachteten VCBS-Servern gemessen. Die erste Messreihe gibt das Verhalten eines VCBS-Servers bei der stetigen Zunahme an Benutzern wieder. Die zweite Messreihe untersucht die Auswirkung einer steigenden Benutzerzahl bei der Verwendung eines weiteren VCBS-Servers. Beide Messreihen verwenden wiederum die zuvor vorgestellte Simulation des Benutzerverhaltens und den Dienst Echo-Service.

Variation der Nutzeranzahl pro Server

Um die Auswirkungen der Anzahl der Benutzer, die einem VCBS-Server zugewiesen werden, auf die Latenz zu untersuchen, wurde in der ersten Messreihe die Anzahl der Benutzer auf einem VCBS-Server stetig erhöht. Für die Messreihe wurden 20 JMeter-Instanzen verwendet, die die Benutzer simulieren (bis zu 200 Benutzer pro JMeter). Der Ablauf der Messung basiert auf der Standardsimulation des Benutzerverhaltens mit den in Tabelle 7.4 angegebenen Parametern.

Tabelle 7.4: Parameter: Nutzeranzahl pro Server

Parameter	Zeitdauer (s)
Start-Phase S	30
Kontext-Phase C	60
Aktiv-Phase A	240
Passiv-Phase D	30
Nutzungs-Phase U	8970

Während der Messung meldet sich alle 10 Sekunden ein neuer Benutzer bei der VCBS-Middleware an, wird dem betrachteten VCBS-Server zugewiesen und führt das beschriebene Benutzerverhalten durch. Für die Nutzungs-Phase wurde eine Dauer von über zwei Stunden gewählt, damit sich kein Benutzer vor dem Ende der Messung wieder abmeldet.

Die Auswertung der durchgeführten Messreihe ergibt, dass bis zu einer Nutzeranzahl, die zwischen 400 und 450 Benutzern liegt, eine ausreichende Latenz (bis 2000 ms) gewährleistet werden kann. Abbildung 7.14 zeigt den Mittelwert über die entstandenen Latenzen der Kommunikation zwischen den Benutzern und dem betrachteten VCBS-Server. Bei bis zu 290 Benutzern wird eine sehr gute durchschnittliche Latenz von unter 200 ms erreicht. Der betrachtete VCBS-Server hat beispielsweise bei 260 Benutzern nur eine geringe Auslastung bei einer durchschnittlichen Latenz von 81ms. Ab etwa 300 Benutzern steigt die Last auf dem Server an. Bei 310 Benutzern besteht noch eine mäßige Auslastung mit einer durchschnittlichen Latenz von 291ms, bei 350 Benutzern besteht etwa eine mittlere Auslastung mit einer durchschnittlichen Latenz von 658ms und bei 370 Benutzern eine hohe Auslastung mit einer durchschnittlichen Latenz von 1082ms. Bei 450 Benutzern ist eine sehr hohe Auslastung des Servers mit einer durchschnittlichen Latenz von 2417ms erreicht.

Ein VCBS-Server ist somit in der Lage, eine Benutzeranzahl von bis zu etwa 400 Benutzer zu unterstützen, ohne dass die dabei entstehende Latenzen zu groß werden. Um große Benutzeranzahlen unterstützen zu können, verwendet die VCBS-Middleware eine Lastverteilung durch den Einsatz verschiedener VCBS-Server, auf die die angemeldeten Benutzer aufgeteilt werden. Diese implementierte Lastverteilung

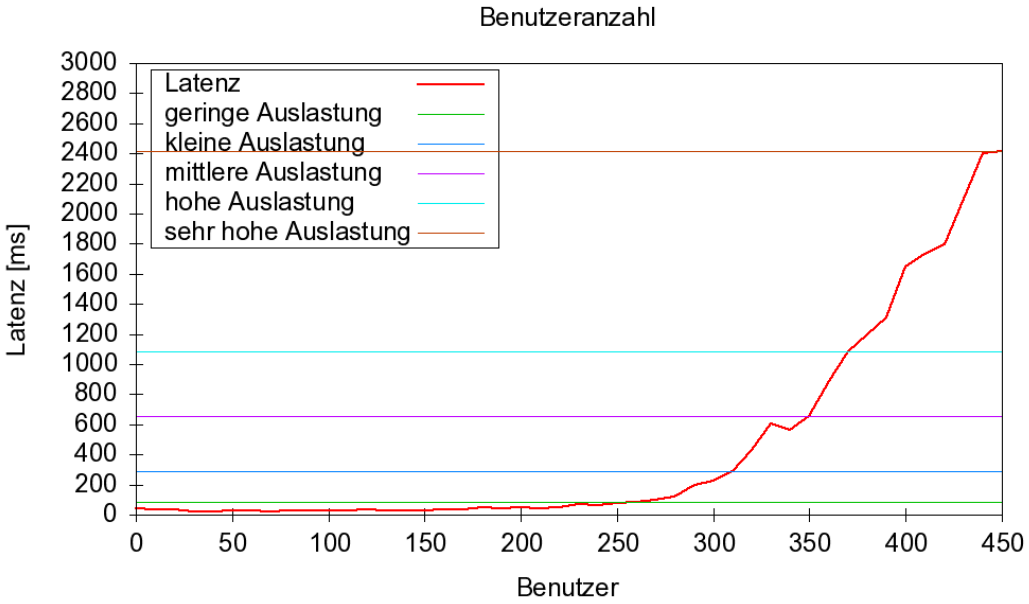

Abbildung 7.14: Latenzen der Kommunikation zwischen den Benutzern und einem VCBS-Server bei steigender Benutzeranzahl

sollte bei einer Verdopplung der Serveranzahl mindestens die Anzahl der möglichen Benutzer verdoppeln. Im Folgenden wird daher die Verwendung von zwei Servern anstelle von Einem untersucht.

Verteilung der Benutzer auf verschiedene Server

Zum Zweck der Lastverteilung sieht die VCBS-Middleware den Einsatz verschiedener Instanzen des VCBS-Servers vor, auf die die angemeldeten Benutzer verteilt werden. Zur Untersuchung der Wirksamkeit der von der VCBS-Middleware durchgeführten Lastverteilung und der dadurch erzielten Latenzverbesserungen wurde in einer weiteren Messreihe die Verwendung eines VCBS-Servers der Verwendung von zwei VCBS-Servern entgegengestellt. Die Messreihe basiert auf der Standardsimulation des Benutzerverhaltens mit den in Tabelle 7.5 gegebenen Parametern.

Tabelle 7.5: Parameter: Nutzerverteilung auf verschiedene Server

Parameter	Zeitdauer (s)
Start-Phase S	30
Kontext-Phase C	60
Aktiv-Phase A	240
Passiv-Phase D	30
Nutzungs-Phase U	8970

Es wurden 20 JMeter-Instanzen verwendet, die die Benutzer simulieren. Die Benutzeranzahl wurde über die Messungen variiert, um das gesamte Spektrum von geringer bis zu sehr hoher Auslastung des Servers zu betrachten. Die Messreihe zeigt, dass sich eine durchschnittliche Verbesserung der Latenz allein durch die Verwendung eines zweiten Servers erreichen lässt. Abbildungen 7.15 und 7.16 zeigen typische Verläufe der gemessenen Latenzen, die den Vergleich zwischen der Verwendung von einem und zwei Servern bei unterschiedlichen Auslastungen, bedingt durch unterschiedlich hohe Benutzerzahlen darstellen.

Die steigenden Kurven zu Beginn der Messungen ergeben sich dadurch, dass sich die Benutzer nacheinander anmelden. Das zu erkennende Plateau zeigt jeweils die Latenzen, für die Zeiten, in denen alle

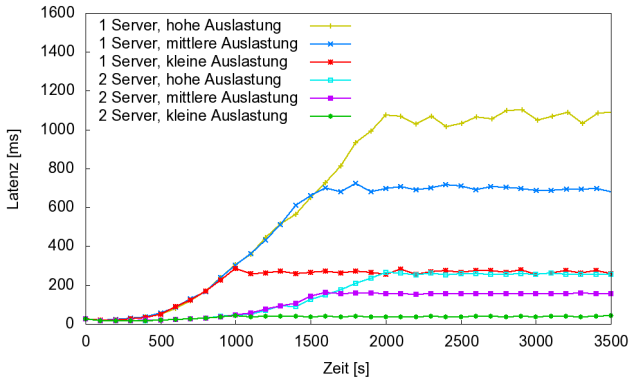

Abbildung 7.15: Durchschnittliche Latenzen bei kleiner bis hoher Auslastung

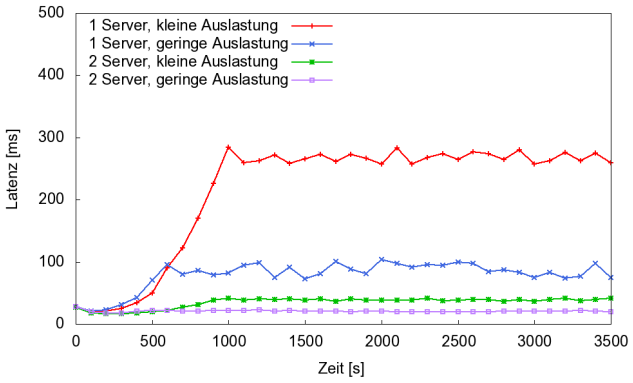

Abbildung 7.16: Durchschnittliche Latenzen bei geringer bis kleiner Auslastung

Benutzer angemeldet sind und den zugewiesenen VCBS-Server nutzen. Die Abbildungen zeigen den Unterschied in der Latenz bei der Verwendung von einem bzw. zwei Servern für vier verschiedene Konfigurationen. Als Referenz dienen die Latenzeiten bei der Verwendung von nur einem VCBS-Server.

Bei einer durch eine geringe Benutzeranzahl erzeugten geringen Auslastung, mit Latenzen von durchschnittlich 87,1ms, liegen die Latenzen bei der Verwendung von zwei Servern bei 20,9ms. Bei einer kleinen Benutzeranzahl und kleiner Auslastung mit Latenzen von 267ms im Durchschnitt, liegen die Latenzen von zwei Servern bei 38,5ms. Bei mittlerer Benutzeranzahl und mittlerer Auslastung mit durchschnittlichen Latenzen von fast 700ms (696,04ms) bei einem Server, liegen bei zwei Servern die Latenzen für die Benutzer unter 200ms (156,47ms). Bei hoher Benutzeranzahl mit hoher Auslastung und Latenzen von durchschnittlich über 1000ms (1062,47ms) wird durch den zweiten Server eine Latenz-Reduktion auf 256,88ms erreicht. Sogar bei einer durch viele Benutzer verursachten sehr hohen Auslastung mit Latenzen von über 2000ms (2893,2ms) bei einem Server, liegen die Latenzen bei zwei Servern noch unter 1000ms (853ms) (siehe auch Tabelle 7.6).

Tabelle 7.6: Serververteilung

Benutzeranzahl und Auslastung bei einem Server	1 Server ∅ Latenzen [ms]	2 Server ∅ Latenzen [ms]	∅ Verbesserung
Gering	87,1	20,9	76 %
Klein	267	38,5	85,5 %
Mittel	696,04	156,47	77,5 %
Hoch	1062,47	256,88	75,8 %
Sehr Hoch	2893,2	853	70,5 %
Gesamte ∅ Verbesserung			77,06 %

Die Verwendung mehrerer VCBS-Server und die Aufteilung der Benutzer auf diese Server zeigt eine deutliche Verbesserung der gemessen Latenz. Die gesamte durchschnittliche Verbesserung bei der Verwendung von zwei Servern, im Vergleich zu einem Server, liegt bei 77% und übertrifft somit sogar die benötigt Verbesserung von 50 % erheblich. Die Anfangs getroffene Annahme, dass durch die von der VCBS-Middleware durchgeführte Verteilung der Benutzer auf verschiedene VCBS-Server die Middleware skalierbar ist und somit auch für große Benutzerzahlen eingesetzt werden kann, konnte durch die durchgeführten Messung im vollen Umfang bestätigt werden.

7.3.3 Einfluss der Dienstnutzung

Um die Auswirkungen der Dienstverwendung auf die Latenz zu untersuchen, wurden zwei weitere Messreihen durchgeführt. Die erste Messreihe untersucht die grundlegenden Auswirkungen der Verwendung eines externen Dienstes auf die Latenz. In der zweiten Messreihe wird der Einfluss verschiedener Dienstarten betrachtet. Dafür wird ein einfacher Dienst mit einem komplexen Dienst verglichen.

Einfluss der Verwendung eines externen Dienstes

Zur Untersuchung des Einflusses der Verwendung eines externen Dienstes durch die Benutzer der VCBS-Middleware auf die Latenz wurde wiederum ein ausgewählter VCBS-Server betrachtet. Für die Messreihe wurde der Echo-Dienst verwendet, der einen kompletten Satz der Kontextparameter (Basis-xgdl) von einem Benutzer erhält und diese als Servicedaten wieder zurücksendet. Die Messreihe ist in zwei verschiedene Phasen unterteilt. In der ersten Phase sind alle Benutzer angemeldet, verwenden jedoch keine Dienste. In der zweiten Phase fangen die Benutzer im Abstand von 10 Sekunden an, den Echo-Dienst zu aktivieren, bis alle Benutzer den Dienst verwenden. Die Messreihe wurde mit 20 JMeter-Instanzen durchgeführt, die die Benutzer simulieren. Die Messreihe basiert auf der Standardsimulation des Benutzerverhaltens mit den in Tabelle 7.7 gegebenen Parametern.

Tabelle 7.7: Parameter: Einfluss der Dienstverwendung

Parameter	Zeitdauer (s)
Start-Phase S	2100
Kontext-Phase C	60
Aktiv-Phase A	4000
Passiv-Phase D	30
Nutzungs-Phase U	4000

Nachdem alle Benutzer eingeloggt sind, beginnt Phase 1 der Messung, in der zwar alle Benutzer aktiv sind, jedoch noch kein Benutzer einen Dienst verwendet. Dabei zeigt sich, dass die Benutzeranzahl keinen Einfluss auf die Latenz hat, solange die Benutzer zwar den VCBS-Server verwenden, jedoch keine Dienste nutzen (siehe Abbildung 7.17 und Spalte "Keine Dienstnutzung" in Tabelle 7.8).

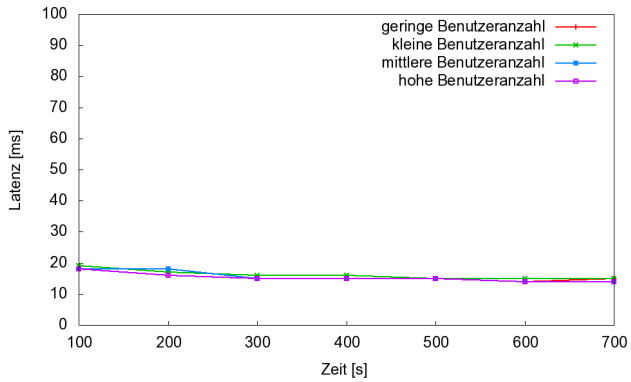

Abbildung 7.17: Phase 1: Keine Dienstnutzung bei unterschiedlicher Benutzeranzahl

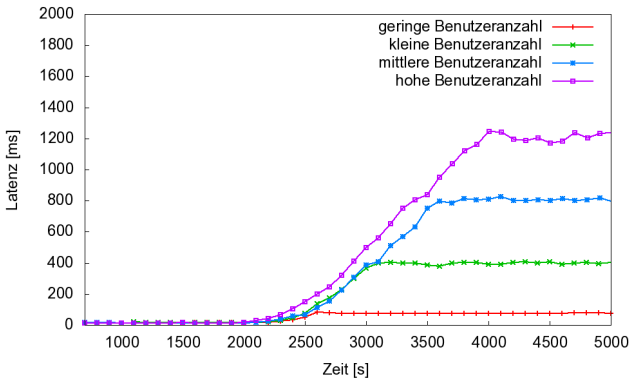

Abbildung 7.18: Phase 2: Dienstnutzung bei unterschiedlicher Benutzeranzahl

In Phase 2 aktivieren die Benutzer nacheinander alle den externen Echo-Dienst, bis schließlich alle Benutzer den Dienst verwenden. Dabei zeigt sich, dass der Einfluss der Dienstverwendung auf die Latenz stark von der Anzahl der Benutzer abhängt, die auf dem VCBS-Server aktiv sind sowie von der Anzahl der Benutzer, die einen Dienst verwenden. Abbildung 7.18 zeigt den Verlauf der durchschnittlichen La-

tenzzeiten bei verschiedenen Benutzeranzahlen. Die Spalte "Volle Dienstnutzung" in Tabelle 7.8 zeigt die durchschnittlichen Latenzen, für den Fall, dass alle Benutzer bereits den Dienst verwenden (in Abbildung 7.18 ab 4100s). Die Spalte "∅ Zunahme" in Tabelle 7.8 gibt die durchschnittliche Zunahme der Latenz bei Erhöhung der Benutzeranzahl für die volle Dienstnutzung aller Benutzer an.

Tabelle 7.8: Dienstverwendung

Benutzeranzahl	Keine Dienstnutzung ∅ Latenzen [ms]	Volle Dienstnutzung ∅ Latenzen [ms]	∅ Zunahme
Gering	15,5	75,3	
Klein	15,7	397,2	321,9
Mittel	14,6	797,7	400,5
Hoch	14,6	1205,6	407,9

Solange ein Benutzer die VCBS-Middleware verwendet, jedoch keinen Dienst aktiviert, werden seine virtuellen Kontextinformationen (virtuelle Parameter) an einen VCBS-Server übertragen und bearbeitet, sowie jeweils passende Dienste zur Nutzung angeboten. Diese grundlegenden Aktionen, die von der VCBS-Middleware durchgeführt werden, erzeugen nur Kommunikationsnachrichten von geringer Größe. Die versendeten Kommunikationsnachrichten sind wesentlich größer, sobald ein Benutzer Dienste verwendet. Vor allem die Servicedaten, die vom externen Dienst erzeugt werden und über den VCBS-Server an die Benutzer weitergeleitet werden, können ein vergleichsweise erhebliches Datenvolumen haben. Dadurch erhöht sich die Last auf dem VCBS-Server sowie das Kommunikationsaufkommen zwischen dem VCBS-Client des Benutzers und dem VCBS-Server. Die Weiterleitung der gefilterten virtuellen Kontextinformationen an die Dienste hat zwar keinen Einfluss auf die Kommunikation zwischen Client und Server, erhöht jedoch auch die auf dem VCBS-Server entstehende Last.

Der Unterschied in der Latenz zwischen der Situation ohne Dienstnutzung und derjenigen mit Dienstnutzung liegt zum einen in den zusätzlichen Aufgaben des VCBS-Servers selbst begründet. Diese sind die Filterung und Weiterleitung der virtuellen Parameter für die Dienste und die Sammlung und Bereitstellung der Servicedaten für den Benutzer. In dieser Messreihe wurde jedoch auf eine Filterung verzichtet, um diesen Einflussfaktor vernachlässigen zu können. Zum anderen ist der Unterschied begründet in der Erhöhung des Kommunikationsaufkommens zwischen Client und Server durch den zusätzlichen Aufruf von *getServiceData* durch den VCBS-Client (siehe auch Kapitel 6.1.3) und besonders der damit verbundenen Weiterleitung der Servicedaten an den Client.

Einfluss der Dienstart

Zur weiteren Untersuchung des Einflusses externer Dienste auf die Latenz werden zwei unterschiedliche Variationen des Echo-Dienstes betrachtet. Der **einfache Dienst**, der keine Vorbedingungen spezifiziert, nur einen virtuellen Parameter erwartet und keine Servicedaten zurücksendet, und der **komplexe Dienst**, der eine Reihe von Vorbedingungen (UND-Verknüpfung von fünf Parametern) spezifiziert, eine volle Basis-xgdl als Parameter erwartet und zufällige Daten (zehn UUIDs in String-Form) generiert, die er, immer wenn er *ClientData* erhält, als Servicedaten zum VCBS-Server zurücksendet. Die Messreihe wurde mit einer unterschiedlichen Anzahl von Benutzern auf einem VCBS-Server durchgeführt. Die Laufzeit der Messungen beträgt jeweils eine Stunde. Die Messreihe wurde wieder mit 20 JMeter-Instanzen durchgeführt und basiert auf der Standardsimulation des Benutzerverhaltens mit den in Tabelle 7.9 gegebenen Parametern.

Die dargestellten Messungen (siehe Abbildung 7.19 und 7.20) zeigen jeweils den Verlauf der Latenz, nachdem sich bereits alle Benutzer bei der VCBS-Middleware angemeldet haben und dem betrachteten VCBS-Server zugewiesen wurden. Zu Beginn der Messungen aktivieren die Benutzer zunächst nacheinander alle den jeweils angebotenen Dienst.

Tabelle 7.9: Parameter: Einfluss der Dienstart

Parameter	Zeitdauer (s)
Start-Phase S	30
Kontext-Phase C	60
Aktiv-Phase A	240
Passiv-Phase D	30
Nutzungs-Phase U	3600

Die Messungen zeigen, dass bei einer geringen Benutzeranzahl die Art der Dienste so gut wie keine Auswirkungen auf die Latenz hat und sich durch die Verwendung der Dienste im Durchschnitt nur marginal eine Erhöhung der Latenzen um 0,4ms (1,6 %) ergibt. Die durchschnittliche Latenz liegt für beide Dienstarten bei 24ms (siehe Abbildung 7.19 und Tabelle 7.10 für geringe Benutzeranzahl).

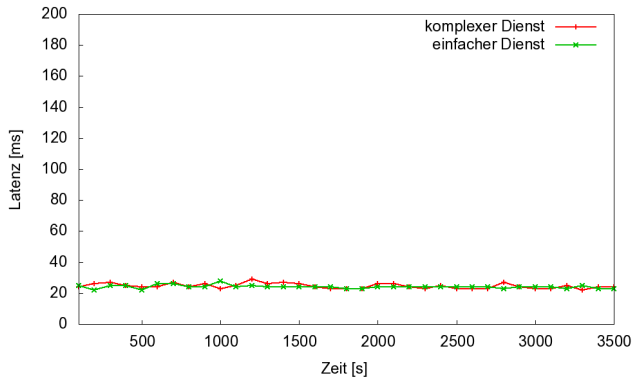

Abbildung 7.19: Vergleich einfacher und komplexer Dienste bei geringer Benutzeranzahl

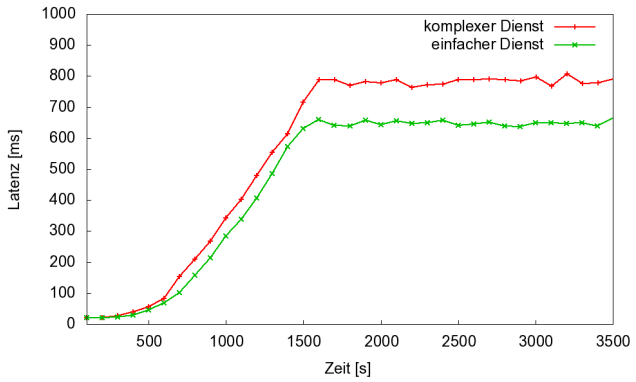

Abbildung 7.20: Vergleich einfacher und komplexer Dienste bei mittlerer Benutzeranzahl

Sind mehr Benutzer auf dem betrachteten VCBS-Server aktiv, dann kann eine Erhöhung der entstehenden Latenzen beobachtet werden, die um 132,7ms (20,8 %) größer ist wenn alle Benutzer nicht den einfachen sondern den komplexen Dienst verwenden. Die durchschnittlichen Latenzen betragen im Schnitt 637,1ms für den einfachen und 759,6ms für den komplexen Dienst (siehe Abbildung 7.20 und Tabelle 7.10 für mittlere Benutzeranzahl).

Tabelle 7.10: Diensteinfluss

Benutzeranzahl	Einfacher Dienst ∅ Latenzen [ms]	Komplexer Dienst ∅ Latenzen [ms]	∅ Differenz [ms]
Gering	24,03	24,43	0,4 (1,6%)
Mittel	637,1	769,6	132,7 (20,8%)

Der Vergleich der beiden Diensttypen zeigt, dass bei einer geringen Benutzeranzahl die Verwendung der Dienste - ob einfache oder komplexe Dienste - keinen Einfluss hat. Bei geringer Auslastung des Servers erhöhen weder das komplexere Dienst-Matching noch die erhöhte Serviedatenmenge die Latenzen der Client-Server Kommunikation. In diesem Fall bestätigt sich die getroffene Annahme bezüglich der Verwendung verschiedenartiger Dienste Der Einfluss der Dienstart wirkt sich nicht im großen Maße auf die Latenz aus.

Bei einer mittleren Benutzeranzahl steigt die Latenz sowohl bei der Verwendung des einfachen Dienstes als auch bei der Verwendung des komplexen Dienstes an. Unter mittlerer Auslastung machen sich

die Unterschiede der Dienste somit bemerkbar. Der gemessene Unterschied zwischen dem einfachen und dem komplexen Dienst liegt beispielsweise bei durchschnittlich 132ms (20,8 %). Die höhere Latenz liegt dabei in erster Linie an den Servicedaten, die vom Dienst über den VCBS-Server an den VCBS-Client übertragen werden. Diese Daten werden im VCBS-Server mittels einer einfachen Pipeline zwischengespeichert und für aktive Dienste an den Benutzer weitergegeben.

Das bestätigt auch der Vergleich der Resultate dieser Messreihe mit der vorherigen Messreihe zum Einfluss der Verwendung eines externen Dienstes. Vergleicht man den in der vorherigen Messreihe verwendeten Echo-Dienst mit dem komplexen Dienst dieser Messreihe so stellt man fest, dass die Latenz bei Verwendung des Echo-Dienstes um durchschnittlich 76,7ms höher ist als bei Verwendung des komplexen Dienstes. Beide Dienste haben dieselben *Required Parameter*, sie erwarten einen kompletten Satz virtueller Parameter (Basis-xgdl) vom VCBS-Server. Sie unterscheiden sich jedoch in der Menge der Servicedaten und in den durch die *Precondion Parameter* spezifizierten Vorbedingungen. Die im VCBS-Server anhand der *Precondition Parameter* durchgeführte Überprüfung der Dienste auf Gültigkeit für die aktuelle Situation des Benutzers wurde nur für den komplexen Dienst durchgeführt, jedoch nicht für den Echo-Dienst, da dieser keine Vorbedingungen spezifiziert. Die Durchführung dieser Überprüfung zeigt somit keine negativen Auswirkungen auf die Performance des VCBS-Servers. Der Echo-Dienst sendet deutlich mehr Servicedaten, da er immer alle empfangenen Daten als Servicedaten wieder an den VCBS-Server zurücksendet. Der komplexe Dienst sendet jeweils nur zehn Strings. Die erhöhte Latenz lässt sich also auf die größere Servicedatenmenge beim Echo-Dienst zurückführen.

7.4 Fazit

Der entwickelte Prototyp der VCBS-Middleware wurde in zwei unterschiedlichen Evalutationsumgebungen ausgewertet. Im ersten Szenario, das durch den Einsatz real existierender Spiele und konkreter Testdienste ein realistisches Setting darstellt, konnte die erfolgreiche Realisierung der funktionalen Anforderungen gezeigt werden. Der Informationsaustausch zwischen den Testspielen und zwei verschiedenen Internet-Applikationen (Community-Portal und Web-Applikation) ist mit der VCBS-Middleware erfolgreich umgesetzt worden (siehe Abschnitt 7.2.1), ebenso wie die Umsetzung des Konzepts der virtuell kontextbasierten Dienste (siehe Abschnitt 7.2.2). Die Funktionstests zeigen auch die erfolgreiche Integration der externen Dienste über das in das Benutzerinterface des Spiels integrierte Service User Interface.

Im zweiten Szenario wurde der entwickelte Prototyp der VCBS-Middleware in einem Simulationstestbed deployed, um Messungen zur Effizienz der Middleware durchzuführen. Die durchgeführten Messungen betrachteten die entstehenden Latenzen der Kommunikation zwischen dem VCBS-Client und dem VCBS-Server. Die durchgeführten Messreihen ermöglichen neben einer Untersuchung der grundlegenden, das Kommunikationsverhalten steuernden Parameter, Untersuchungen der VCBS-Middleware hinsichtlich der Lastverteilung auf mehrere verteilte Server sowie Untersuchungen des Einflusses unterschiedlicher Benutzeranzahlen und verschiedener Diensttypen. Als Ergebnis der Messreihen des zweiten Evaluationsszenarios zeigt sich, dass die VCBS-Middleware effizient arbeitet und dass der gewählte Skalierungsansatz erfolgreich ist (siehe Abschnitt 7.3.2). Der Prototyp ist stabil und zeigt im Allgemeinen das erwartete Verhalten.

Der Einfluss der externen Dienste wirkt sich jedoch in manchen Fällen stärker aus als erwartet. Die Betrachtung des Einflusses der Dienstnutzung auf die Latenz (siehe Abschnitt 7.3.3) zeigt, dass für den Prototyp der VCBS-Middleware noch weiteres Verbesserungspotenzial besteht. Bei Verwendung komplexerer Dienste erhöht sich die gemessene Latenz. Die höhere Latenz liegt dabei in erster Linie an den Servicedaten, die vom Dienst über den VCBS-Server an den VCBS-Client übertragen werden. Diese Daten werden im VCBS-Server mittels einer einfachen Pipeline zwischengespeichert und für aktive Dienste an den Benutzer weitergegeben. Diese Datenweiterleitung von der Daten-Pipeline im VCBS-Server an den VCBS-Client ist in der Implementierung der Middleware an das Dienst-Matching gekoppelt. Dadurch

wird eine zusätzliche Last auf dem Server erzeugt, wenn ein Client die für ihn zwischengespeicherten Servicedaten abruft. Diese enge Kopplung ist für die Funktionalität der Middleware jedoch nicht nötig. Alternativ kann eine einfache Übersicht (beispielsweise durch eine Liste) der aktiven Dienste des Benutzers für die Servicedatenweiterleitung verwendet werden.

Auch die Betrachtung des Einflusses der Benutzeranzahl auf die Latenz (siehe Abschnitt 7.3.2) zeigt, dass für den Prototyp der VCBS-Middleware noch weiteres Verbesserungspotenzial besteht. Die Annahme der Kommunikationsaufrufe von den VCBS-Clients wird im Prototyp des VCBS-Servers durch eine einfache Queue realisiert. Die Analyse der Messungen hat gezeigt, dass diese Queue für die entstehenden Verzögerungen bei steigendem Kommunikationsaufkommen, wie es durch die Erhöhung der Benutzeranzahl auf dem VCBS-Server entsteht, verantwortlich ist und somit die Erhöhung der gemessenen Latenzen maßgeblich verursacht. Alternativ kann das Queueing der Benutzeranfragen beim VCBS-Server verbessert werden, indem beispielsweise andere Queues (die zum Beispiel mit Prioritäten arbeiten) oder verschiedene Queues für verschiedene Kommunikationsaufrufe verwendet werden. Eine andere Alternative ist die Implementierung einer anderen Kommunikationstechnologie, die besser auf die Kommunikation der Middleware zugeschnitten werden kann, beispielsweise auf Basis einer Netzwerk-Bibliothek.

Anhand der durchgeführten Evaluation konnten die Validität des in der Arbeit entwickelten Konzeptes der virtuell kontextbasierten Dienste und die Umsetzbarkeit einer Architektur zur Realisierung des Informationsaustauschs auf Grundlage von kontextbasierten Diensten gezeigt werden. Eine weitere Entwicklung des implementierten Prototypen zur Verbesserung von Effizienz und Performance der VCBS-Middleware kann vor allem durch eine Verbesserung der Datenweiterleitung der Servicedaten vorgenommen werden, beispielsweise durch eine bessere Entkopplung. Auch durch eine Verbesserung des Queueing der Benutzeranfragen beim VCBS-Server oder durch den Einsatz einer auf die Middleware zugeschnittenen Kommunikationstechnologie können Effizienz und Performance verbessert werden.

8 Zusammenfassung und Ausblick

Das abschließende Kapitel gibt eine Zusammenfassung der Hauptbeiträge der Arbeit. Darauf folgen ein Ausblick und die Identifikation von Ansatzpunkten, die sich durch die vorliegende Arbeit ergeben haben.

8.1 Zusammenfassung

Die vorliegende Arbeit hatte zum Ziel, eine Benutzerunterstützung in virtuellen Welten mittels situationsbezogener Dienste zu ermöglichen, sowie einen Informationsaustausch zwischen virtuellen Welten und anderen Internet-Applikationen zu ermöglichen, um externe Dienste geeignet an virtuelle Welten anbinden zu können.

Dieses Ziel wurde in der Arbeit auf Basis des virtuellen Kontexts des Benutzers erreicht. Der virtuelle Kontext ist Grundlage für eine situationsbezogene Anpassung von Diensten und Dienstangebot in virtuellen Welten und ermöglicht einen kontrollieren Informationsaustausch. Mittels der Erfassung virtueller Kontextinformationen, wie beispielsweise der Lokation des virtuellen Charakters oder dessen Ausrüstung, kann die aktuelle virtuelle Situation des Benutzers ermittelt werden, wodurch die Anpassung der Dienste und des Dienstangebots entsprechend der Situation des Benutzers ermöglicht wird. Durch die automatische Anpassung des Dienstangebots muss der Benutzer nicht selbst aktiv nach Diensten suchen, sondern kann immer auf ein seiner aktuellen Situation entsprechendes Dienstangebot zurückgreifen. Dies hat den Vorteil, dass das Dienstangebot überschaubar bleibt und der Benutzer nicht manuell die relevanten Dienste heraussuchen muss. Darüber hinaus ermöglicht das in dieser Arbeit realisierte Verfahren des Informationsaustauschs die Weitergabe von in einem Multiplayer Online Game erfassten Informationen an externe Dienstanbieter, was eine weitere Nutzung der in-game Informationen ermöglicht. Die Trennung von Kontexterfassung und Kontextnutzung erlaubt eine Beteiligung der Community bei der Entwicklung von Diensten, da Dienste von den Benutzern beziehungsweise der Community selbst entwickelt werden können.

Diese benutzerzentrierte und situationsbasierte Verknüpfung zwischen virtueller Welt und externen Diensten und die Realisierung des Informationsaustauschs wird in der Arbeit durch die Konzeption und Realisierung einer Middleware-Architektur, basierend auf virtuellem Kontext, ermöglicht (VCBS-Middlware). Die VCBS-Middlware stellt dabei für den Benutzer eine Verbindung zu Informationen und Funktionen externer Dienste her. Sie schafft eine generische Verknüpfung von Spiel und externen Internet-Applikationen im Spielumfeld und ermöglicht einen Informationsfluss in beide Richtungen, die Verwendung von in-game Informationen außerhalb eines Spiels und die Bereitstellung von out-game Daten (Servicedaten) innerhalb eines Spiels.

Die folgenden Beiträge der Arbeit schaffen die Voraussetzungen zur Verwendung von virtuellem Kontext und zur Realisierung des Informationsaustauschs zwischen virtuellen Welten und anderen Internet-Applikationen:

Eine umfangreiche Analyse virtueller Welten im Allgemeinen und von Multiplayer Online Games im Speziellen wurde durchgeführt. Basierend auf der Untersuchung der Interaktionsmöglichkeiten in virtuellen Welten wurde ein generisches Interaktionsmodell entwickelt. Die Übertragbarkeit des Kontextkonzepts auf virtuelle Welten wurde untersucht und das Konzept des virtuellen Kontexts definiert. Virtuelle Parameter wurden als Grundlage für die Verwendung von virtuellem Kontext identifiziert. Basierend auf den virtuellen Parametern wurde eine Beschreibungssprache für virtuellen Kontext entwickelt, die eXtensible Game Description Language (xgdl).

Für die Realisierung der Benutzerunterstützung in virtuellen Welten wurde das Konzept der virtuell kontextbasierten Dienste (Virtual Context Based Services - VCBS) entwickelt. Virtuell kontextbasierte

Dienste ermöglichen das Angebot und die Bereitstellung von Diensten in virtuellen Welten basierend auf der virtuellen Situation des Benutzers. Durch die virtuell kontextbasierten Dienste wird die Grundlage für einen kontrollierten Informationsaustausch zwischen einer virtuellen Welt und anderen Internet-Applikationen geschaffen und die Verwendung zusätzlicher externer Dienste in virtuellen Welten ermöglicht.

Zur Umsetzung des Informationsaustauschs wurde ein Middleware-Ansatz entwickelt. Weiterhin stellt die Arbeit neue Konzepte für die Integration und Darstellung von Diensten in virtuellen Welten und die aktive Beteiligung der Community vor. Darüber hinaus werden die durch die Möglichkeiten des Informationsaustauschs entstehenden Sicherheitsanforderungen in den Bereichen Privacy und Cheating adressiert.

Die entwickelten Konzepte wurden prototypisch implementiert und evaluiert. Die funktionale Evaluation der entwickelten VCBS-Middleware wurde in einem realistischen Szenario anhand von konkreten Beispieldiensten sowie unter Verwendung zweier kommerzieller Multiplayer Online Games durchgeführt. Zusätzlich wurde die VCBS-Middleware hinsichtlich ausgewählter Systemeingenschaften, mittels einer Simulation großer Nutzeranzahlen in einem geeigneten Testbed, evaluiert.

8.2 Ausblick

Es gibt verschiedene Ansatzpunkte für weitere Arbeiten an der VCBS-Middleware: Die prototypische Implementierung der Middleware-Architektur kann in verschiedenen Bereichen noch verbessert werden. Dazu zählen vor allem eine Verbesserung der eingesetzten Verfahren zur Kommunikation, sowie eine Entkopplung der Servicedatenweiterleitung von der Kontextverarbeitung. Die Anbindung aktuell verfügbarer virtueller Welten wird über Plug-Ins realisiert. Deshalb kann die Umsetzung weiterer Plug-Ins für verschiedene virtuelle Welten ein weiterer Schritt sein.

Die in der Arbeit entwickelte Beschreibung von virtuellem Kontext basierend auf virtuellen Parametern kann die Grundlage für eine generische Schnittstelle zukünftiger virtueller Welten darstellen. Eine einheitliche Schnittstelle würde die einfache Nutzung von virtuell kontextbasierten Diensten ermöglichen und den virtuellen Welten, die diese Schnittstelle unterstützen, den Mehrwert der externen Zusatzdienste erschließen. Zusätzlich wird eine aktive Beteiligung der Community ermöglicht. Die entsprechende Standardisierung einer generischen Schnittstelle virtueller Welten basierend auf virtuellen Parametern wäre ein erster Schritt zu einer solchen einheitlichen Schnittstelle und ein weiterer Schritt dieser Arbeit.

Die Entwicklung von Diensten durch die Community ist ein wichtiger Aspekt der Arbeit. Durch die in der Arbeit konzipierten und umgesetzten Verfahren entsteht die Möglichkeit einer aktiven Beteiligung der Community. Somit können die Benutzer einer virtuellen Welt spefizische Dienste, entsprechend ihres jeweiligen Bedarfs, umsetzten. Die so entstehenden Zusatzdienste zu untersuchen und die Verwendung der Dienste durch die Benutzer zu evaluieren, sind interessante Aspekte, die sich durch die vorliegende Arbeit ergeben. Ein entsprechender weiterer Schritt wäre, die VCBS-Middleware in einer groß angelegten Benutzerstudie einzusetzen oder sie in Kooperation mit mindestens einem Anbieter einer virtuellen Welt einer großen Benutzerbasis anzubieten.

Über die in dieser Arbeit fokussierten Multiplayer Online Games hinaus, stellen virtuelle Welten kollaborative Applikationen dar, die in unterschiedlichen Bereichen, wie beispielsweise dem interdisziplinären wissenschaftlichen Arbeiten, eingesetzt werden. Gerade auch in diesem Bereich ist die Integration von kontextbasierten Zusatzdiensten zur Benutzerunterstützung ein wertvoller Beitrag, um virtuelle Welten an die Bedürfnisse ihrer Benutzer anzupassen. Diese Arbeit legt die Grundlagen für einen Einsatz virtuell kontextbasierter Dienste in virtuellen Welten. Ein Einsatz der VCBS-Middleware im Szenario des kollaborativen wissenschaftlichen Arbeitens wäre ein weiterer Schritt.

Literaturverzeichnis

[ADOB98] Gregory D. Abowd, Anind K. Dey, Robert Orr, and Jason Brotherton. Context-awareness in wearable and ubiquitous computing. In *Journal of Virtual Reality*, volume 3, pages 200–211. Springer-Verlag, 1998. 15

[Ald01] Liso. Aldo. Software maintainability metrics model: An improvement in the coleman-oman model. *Crosstalk - The Journal of Defense Software Engineering*, 2001. 114

[AT06] Marios Assiotis and Velin Tzanov. A distributed architecture for mmorpg. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 4, New York, NY, USA, 2006. ACM. 33

[Bar90] Richard A. Bartle. Interactive multi-user computer games, June 1990. report commissioned by British Telecom on the state of the on-line multi-player game industry at the time. 26

[Bar96] Richard A. Bartle. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of Virtual Environments*, 1(1), 1996. 96

[BB04] Barry Brown and Marek Bell. Cscw at play: 'there' as a collaborative virtual environment. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 350–359, New York, NY, USA, 2004. ACM. 1, 24, 38

[BBC97] P. J. Brown, J. D. Bovey, and Xian Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64, Oct. 1997. 13, 15

[BCL+04] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. The effects of loss and latency on user performance in unreal tournament 2003®. In *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pages 144–151, New York, NY, USA, 2004. ACM. 33

[BD03] Louise Barkhuus and Anind Dey. Is context-aware computing taking control away from the user? three levels of interactivity examined. In *In Proceedings of Ubicomp 2003*, pages 150–156, Seattle, USA, October 12-15 2003. Springer. 19

[BDR07] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007. 13, 20, 22, 24, 84

[Ber01] Y.W. Bernier. Latency compensating methods in client/ server in-game protocol design and optimization. In *Proceedings of the Game Developers Conference*, February 2001. 31, 33

[BKV06] Jean-Sébastien Boulanger, Jörg Kienzle, and Clark Verbrugge. Comparing interest management algorithms for massively multiplayer games. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 6, New York, NY, USA, 2006. ACM. 31

[Bro96] P. J. Brown. The stick-e document: a framework for creating context-aware applications. In *Proceedings of EP'96, Palo Alto*, pages 259–272, June 1996. 13

[BRS02] Daniel Bauer, Sean Rooney, and Paolo Scotton. Network infrastructure for massively distributed games. In *NetGames '02: Proceedings of the 1st workshop on Network and system support for games*, pages 36–43, New York, NY, USA, 2002. ACM. 33

[BVHG09] Paul B. Beskow, Knut-Helge Vik, Pål Halvorsen, and Carsten Griwodz. The partial migration of game state and dynamic server selection to reduce latency. *Multimedia Tools and Applications*, 45(1-3):83–107, 2009. 32

[CC06] Mark Claypool and Kajal Claypool. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, 2006. 27, 32

[CFFS05] Chris Chambers, Wu-chang Feng, Wu-chi Feng, and Debanjan Saha. Mitigating information exposure to cheaters in real-time strategy games. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 7–12, New York, NY, USA, 2005. ACM. 79

[CFJ04] Harry Chen, Tim Finin, and Anupam Joshi. Semantic web in the context broker architecture. In *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, page 277, Washington, DC, USA, 2004. IEEE Computer Society. 19, 22, 79

[CFSS05] Chris Chambers, Wu-chang Feng, Sambit Sahu, and Debanjan Saha. Measurement-based characterization of a collection of on-line games. In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 1–1, Berkeley, CA, USA, 2005. USENIX Association. 31

[Che04] Harry Lik Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, University of Maryland, 2004. 20

[Che07] Jenova Chen. Flow in games (and everything else). *Communications of the ACM*, 50(4):31–34, 2007. 9

[Che09a] Mark Chen. Social dimensions of expertise in world of warcraft players. *Transformative Works and Cultures*, 2, 2009. 5

[Che09c] Mark G. Chen. Communication, coordination, and camaraderie in world of warcraft. *Games and Culture*, 4(1):47–73, January 2009. 56

[CHL06a] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. Game traffic analysis: An MMORPG perspective. *Computer Networks*, 50(16):3002–3023, Nov 2006. 31

[CHL06b] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. How sensitive are online gamers to network quality? *Communications of the ACM*, 49(11):34–38, Nov 2006. 33

[CHL09] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. Effect of network quality on player departure behavior in online games. *IEEE Transactions on Parallel and Distributed Systems*, 20(5):593–606, May 2009. 32

[CK00] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical report, 2000. 14, 15, 24, 79, 96

[Cla05] Mark Claypool. The effect of latency on user performance in real-time strategy games. *Computer Networks*, 49(1):52 – 70, 2005. Networking Issue in Entertainment Computing. 33

[CTC09] Chia-Jung Chan, Ruck Thawonmas, and Kuan-Ta Chen. Automatic storytelling in comics: a case study on world of warcraft. In *CHI '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3589–3594, New York, NY, USA, 2009. ACM. 41

[CXTL02] Wentong Cai, Percival Xavier, Stephen J. Turner, and Bu-Sung Lee. A scalable architecture for supporting interactive games on the internet. In *PADS '02: Proceedings of the sixteenth workshop on Parallel and distributed simulation*, pages 60–67, Washington, DC, USA, 2002. IEEE Computer Society. 33

[DA00] A.K. Dey and G.D. Abowd. Towards a better understanding of context and context-awareness. In *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*, 2000. 14, 15, 17

[DAS01] Anind K. Dey, Gregory D. Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2):97–166, 2001. 20, 21

[DAW98] Anind K. Dey, Gregory D. Abowd, and Andrew Wood. Cyberdesk: a framework for providing self-integrating context-aware services. In *Third International Conference on Intelligent User Interfaces*, page 47 54, San Francisco, California, USA, 1998. ACM Press. 14

[Deb02] E. Debold. Flow with soul: An interview with dr. mihaly csikszentmihalyi. What Is Enlightenment Magazine (Spring-Summer 2002), 2002. 9

[Dey00] Anind Kumar Dey. *Providing architectural support for building context-aware applications*. PhD thesis, Atlanta, GA, USA, 2000. Director-Abowd, Gregory D. 14, 20

[Dey01] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5:4–7, 2001. 16

[DM04] N. Ducheneaut and R.J. Moore. The social side of gaming: a study of interaction patterns in a massively multiplayer online game. In *conference proceedings on computer-supported cooperative work (CSCW 2004)*, 2004. 38, 56

[DMLS04] Vasilios Darlagiannis, Andreas Mauthe, Nicolas Liebau, and Ralf Steinmetz. Distributed maintenance of mutable information for virtual environments. In *Proceedings of 3rd IEEE International Workshop on Haptic Audio Visual Environments and their Applications - HAVE'04*, pages 87–92, Oct 2004. 32

[dRE06] Ricardo Couto A. da Rocha and Markus Endler. Supporting context-aware applications: Scenarios, models and architecture. In *Proc. of the XXIV Brazilian Symposium on Computer Networks (SBRC 2006)*, Curitiba, May 2006. 19

[dREdS08] Ricardo Couto A. da Rocha, Markus Endler, and Thiago Senador de Siqueira. Middleware for ubiquitous context-awareness. In *MPAC '08: Proceedings of the 6th international workshop on Middleware for pervasive and ad-hoc computing*, pages 43–48, New York, NY, USA, 2008. ACM. 24

[DWW05] Matthias Dick, Oliver Wellnitz, and Lars Wolf. Analysis of factors affecting players' performance and perception in multiplayer games. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–7, New York, NY, USA, 2005. ACM. 32

[DYNM06] N. Ducheneaut, N. Yee, E. Nickell, and R.J Moore. Alone together? exploring the social dynamics of massively multiplayer games. In *Conference proceedings on human factors in computing systems (CHI2006)*, 2006. 1, 38, 40

[DYNM07] N. Ducheneaut, N. Yee, E. Nickell, and R. J. Moore. The life and death of online gaming communities: a look at guilds in world of warcraft. In *Proceedings of 25th Annual ACM Conference on Human Factors in Computing Systems (SIGCHI 2007)*, San Jose, California, USA, April 28 - May 03 2007. 38, 80

[Eri02] Thomas Erickson. Some problems with the notion of context-aware computing. *Communications of the ACM*, 45(2):102–104, 2002. 95

[FÖ2] Johannes Färber. Network game traffic modelling. In *NetGames '02: Proceedings of the 1st workshop on Network and system support for games*, pages 53–57, New York, NY, USA, 2002. ACM. 31

[FCFW05] Wu-chang Feng, Francis Chang, Wu-chi Feng, and Jonathan Walpole. A traffic characterization of popular on-line games. *IEEE/ACM Transactions on Networking*, 13(3):488–500, 2005. 31

[FF98] David Franklin and Joshua Flachsbart. All gadget and no representation makes jack a dull environment. In *In AAAI Spring Symposium on Intelligent Environments. AAAI TR*, pages 155–160, 1998. 13

[FKS08] Wu-chang Feng, Ed Kaiser, and Travis Schluessler. Stealth measurements for cheat detection in on-line games. In *NetGames '08: Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 15–20, New York, NY, USA, 2008. ACM. 79

[FRS05] Tobias Fritsch, Hartmut Ritter, and Jochen Schiller. The effect of latency and network limitations on mmorpgs: a field study of everquest2. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–9, New York, NY, USA, 2005. ACM. 33

[FW04] David C. Fallside and Priscilla Walmsley. Xml schema part 0: Primer second edition. Technical report, W3C, 2004. 65

[GD05] Philippe Golle and Nicolas Ducheneaut. Preventing bots from playing online games. *Computers in Entertainment (CIE)*, 3(3), 2005. 79

[GLK+94] Rich Gossweiler, Robert J. Laferriere, Michael L. Keller, Contact R, Y Pausch, and Y Pausch. An introductory tutorial for developing multi-user virtual environments. Technical report, Presence, 1994. 24

[GPZ05] T. Gu, H. K. Pung, and D. Zhang. A service-oriented middleware for building context-aware services. *Elsevier Journal of Network and Computer Applications (JNCA)*, 28(1):1–18, January 2005. 19, 20, 21

[Gör05] Manuel Görtz. *Effiziente Echtzeit-Kommunikationsdienste durch Einbeziehung von Kontexten*. PhD thesis, Technische Universität Darmstadt, 2005. 13, 14, 19

[Gra92] Robert B. Grady. *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall, 1992. 113

[Gwi00] Jacek Gwizdka. What's in the context? In *Computer Human Interaction - The What, Who, Where, Why and How of Context-Awareness*, 2000. 15

[HL01] Jason I. Hong and James A. Landay. An infrastructure approach to context-aware computing. *Human-Computer Interaction*, 16(2):287–303, 2001. 22

[HL04] Shun-Yun Hu and Guan-Ming Liao. Scalable peer-to-peer networked virtual environment. In *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pages 129–133, New York, NY, USA, 2004. ACM. 32

[HL07] Chin-Lung Hsu and Hsi-Peng Lu. Consumer behavior in online game communities: A motivational factor perspective. *Computers in Human Behavior*, Avoiding Simplicity, Confronting Complexity: Advances in Designing Powerful Electronic Learning Environments(Volume 23, Issue 3):1642–1659, May 2007. 34

[HNBR97] Richard Hull, Philip Neaves, and James Bedford-Roberts. Towards situated computing. In *ISWC '97: Proceedings of the 1st IEEE International Symposium on Wearable Computers*, Washington, DC, USA, 1997. IEEE Computer Society. 13

[Hol05] Andreas Holzinger. Usability engineering methods for software developers. *Commun. ACM*, 48(1):71–74, 2005. 113

[HPGH07] Szabolcs Harcsik, Andreas Petlund, Carsten Griwodz, and Pål Halvorsen. Latency evaluation of networking mechanisms for game traffic. In *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 129–134, New York, NY, USA, 2007. ACM. 31

[Hsu05] Wei-Hsiang Hsu. Business model developments for the pc-based massively multiplayer online game (mmogo) industry. Master's thesis, University of Wollongong, 2005. 37

[Hui55] Johan Huizinga. *Homo Ludens: A Study of the Play-Element in Culture*. Beacon Press, Boston, 1955. ISBN 978-0807046814. 25

[ISOa] ISO. Iso 9241-11: Ergonomic requirements for office work with visual display terminals (vdts) - part 11: Guidance on usability. 113

[ISOb] ISO. Iso/iec 9126: Software engineering - product quality. 113

[Jan06] Klaus Jantke. Digitale spiele - eine herausforerung an wissenschaft und entwicklung sowie eine einmalige chance für die wirtschaft. Kongress Multimediatechnik, 2006. 26

[KCC+05] Jaecheol Kim, Jaeyoung Choi, Dukhyun Chang, Taekyoung Kwon, Yanghee Choi, and Eungsu Yuk. Traffic characteristics of a massively multi-player online role playing game. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–8, New York, NY, USA, 2005. ACM. 31

[Äkk05] Miia Äkkinen. Conceptual foundations of online communities. Technical Report W-387, Helsinki School of Economics, September 2005. 34, 36

[Koi03] Elina M.I. Koivisto. Supporting communities in massively multiplayer online role-playing games by game design. In Copier Marinka and Raessens Joost, editors, *Level Up Conference Proceedings*, Utrecht, November 2003. University of Utrecht. 8, 40, 41

[KPM05] Anders Kofod-Petersen and Marius Mikalsen. An architecture supporting implementation of context-aware services. In *Workshop on Context Awareness for Proactive Systems (CAPS 2005)*, pages 31–42, Helsinki, Finland, 2005. HIIT Publications. 15, 24, 79

[LLHL07] Dongman Lee, Mingyu Lim, Seunghyun Han, and Kyungmin Lee. Atlas: A scalable network framework for distributed virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 16(2):125–156, 2007. 24, 32

[Man00] Tony Manninen. Rich interaction in networked virtual environments. In *MULTIMEDIA '00: Proceedings of the eighth ACM international conference on Multimedia (ACM MM 2000)*, pages 517–518, New York, NY, USA, 2000. ACM. 48

[Man02] Tony Manninen. Towards communicative, collaborative and constructive multi-player games. In Mayr&#228 Frans, editor, *Computer Games and Digital Cultures Conference Proceedings*, page 15, Tampere, June 2002. Tampere University Press. 40

[Man03] Tony Manninen. Interaction forms and communicative actions in multiplayer games. *Game Studies - the international journal of computer game research*, 3, Mai 2003. 38, 40

[MBD00] Katherine L. Morse, Lubomir Bic, and Michael Dillencourt. Interest management in large-scale virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 9(1):52–68, 2000. 33

[ME05] Frans Mäyrä and Laura Ermi. Fundamental components of the gameplay experience: Analysing immersion. In *DiGRA 2005 Conference: Changing Views - Worlds in Play*, 2005. 9

[MFW02] Martin Mauve, Stefan Fischer, and Jörg Widmer. A generic proxy system for networked computer games. In *NetGames '02: Proceedings of the 1st workshop on Network and system support for games*, pages 25–28, New York, NY, USA, 2002. ACM. 32, 33

[MN05] Stefan Münz and Wolfgang Nefzger. *HTML Handbuch*. Franzis, 2005. ISBN-10: 3-7723-6654-6 ISBN-13: 978-3-7723-6654-3. 65

[Mos02] Marie-Luise Moschgath. *Kontextabhängige Zugriffskontrolle für Anwendungen im Ubiquitous Computing*. PhD thesis, Technische Universität Darmstadt, 2002. 60, 64

[NC04] James Nichols and Mark Claypool. The effects of latency on online madden nfl football. In *NOSSDAV '04: Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, pages 146–151, New York, NY, USA, 2004. ACM. 33

[NH06] Bonnie Nardi and Justin Harris. Strangers and friends: collaborative play in world of warcraft. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 149–158, New York, NY, USA, 2006. ACM. 1, 38, 39, 56

[Pan05] Gregor Panten. *Internet-Geschäftsmodell Virtuelle Community: Analyse zentraler Erfolgsfaktoren unter Verwendung des Partial-Least-Squares (PLS)-Ansatzes*. DUV, 2005. 34, 35

[Pas98] Jason Pascoe. Adding generic contextual capabilities to wearable computers. In *ISWC '98: Proceedings of the 2nd IEEE International Symposium on Wearable Computers*, page 92, Washington, DC, USA, 1998. IEEE Computer Society. 16

[PPRB06] J. Pauty, D. Preuveneers, P. Rigole, and Y. Berbers. Research challenges in mobile and context-aware service development. In *Workshop on Research Challenges in Mobile and Context-Aware Service Development (FRCSS 2006)*, pages 141–148, Vienna, Austria, April 1 2006. 19

[PVdBW+04] Davy Preuveneers, Jan Van den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koen De Bosschere. *Towards an Extensible Context Ontology for Ambient Intelligence*, volume 3295/2004 of *Lecture Notes in Computer Science*, pages 148–159. Springer Berlin / Heidelberg, ambient intelligence edition, October 2004. 16

[PW02a] Lothar Pantel and Lars C. Wolf. On the impact of delay on real-time multiplayer games. In *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 23–29, New York, NY, USA, 2002. ACM. 33

[PW02b] Lothar Pantel and Lars C. Wolf. On the suitability of dead reckoning schemes for games. In *NetGames '02: Proceedings of the 1st workshop on Network and system support for games*, pages 79–84, New York, NY, USA, 2002. ACM. 89

[Rei99] Brandon Reinhart. Mod authoring for unreal tournament. Technical report, Epic Games, Inc., 1999. 41

[RHC+02] Manuel Román, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74–83, 2002. 21

[Rhe93] Howard Rheingold. *The Virtual Community - Homesteading On The Electronic Frontier*. Reading, Mass. : Addison-Wesley Pub. Co., 1993. 33

[SA04] Aameek Singh and Arup Acharya. Using session initiation protocol to build context-aware voip support for multiplayer networked games. In *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pages 98–105, New York, NY, USA, 2004. ACM. 93, 109

[Sal00] Daniel Salber. Context-awareness and multimodality. In *Colloque sur la multimodalité, IMAG*, Grenoble, Mai 2000. 17

[San03] Derek Sanderson. Everybody needs somebody: Practical advice for encouraging cooperative play in online virtual worlds. *Massively Multiplayer Game Development*, 2003. Editor: Thor Alexander. 56

[SAW94] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proc. First Workshop on Mobile Computing Systems and Applications*, pages 85–90, 8–9 Dec. 1994. 14, 15, 16, 17

[Sch09] Johannes Schmitt. *Anpassungsfähige Kontextbestimmung zur Unterstützung von Kommunikationsdiensten*. PhD thesis, Technische Universität Darmstadt, 2009. 14

[SDM09] Mirko Suznjevic, Ognjen Dobrijevic, and Maja Matijasevic. Hack, slash, and chat: A study of players behavior and communication in mmorpgs. In *Proceedings of Netgames 2009*, 2009. 123

[SE05] Ralf Steinmetz and Klaus Wehrle (Edits.). *Peer-to-Peer Systems and Applications*. Springer, Sep 2005. 31, 32

[SGJ07] Travis Schluessler, Stephen Goglin, and Erik Johnson. Is a bot at the controls?: Detecting input data attacks. In *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 1–6, New York, NY, USA, 2007. ACM. 79

[SJLK04] A. Fleming Seay, William J. Jerome, Kevin Sang Lee, and Robert E. Kraut. Project massive: a study of online gaming communities. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1421–1424, New York, NY, USA, 2004. ACM. 38, 40

[SKH02a] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. Aspects of networking in multiplayer computer games. *The Electronic Library*, 20(2):87–97, 2002. 32, 33, 89

[SKH02b] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. A review on networking and multiplayer computer games. Technical Report 454, Turku Centre for Computer Science, 2002. 24, 25

[Smi05] Jonas Heide Smith. The problem of other players: In-game cooperation as collective action. In *DiGRA 2005 Conference: Changing Views–Worlds in Play*, Vancouver, Canada, 2005. 38, 56

[ST93] Mike Spreitzer and Marvin Theimer. Providing location information in a ubiquitous computing environment. In *SOSP '93: Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 270–283, New York, NY, USA, 1993. ACM. 96

[ST94] B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, Sept.–Oct. 1994. 13, 15

[Str07] Jeff Strain. How to create a successful mmo. Games Convention Developers Conference (GCDC), Leipzig, 2007. 37

[SW05] Penelope Sweetser and Peta Wyeth. Gameflow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, 3(3), 2005. 9

[SZ99] Sandeep Singhal and Michael Zyda. *Networked virtual environments: design and implementation*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999. 24, 33

[SZ04] Katie Salen and Eric Zimmerman. *Rules of Play - Game Design Fundamentals*. MIT Press, 2004. 25, 38, 46

[SZ05] Katie Salen and Eric Zimmerman. *Handbook of Computer Game Studies*, chapter Game Design and Meaningful Play, pages 59–81. MIT Press, 2005. 78

[Tay03] T.L. Taylor. Power games just want to have fun?: instrumental play in a mmog. In *Level Up Conference Proceedings*, November 2003. 2, 40, 96

[TBMM04] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. Xml schema part 1: Structures second edition. Technical report, W3C, 2004. 66

[VGH06] Knut-Helge Vik, Carsten Griwodz, and Pål Halvorsen. Applicability of group communication for increased scalability in mmogs. In Stephane Natkin Adrian David Cheok, Yutaka Ishibashi and Keiichi Yasumoto, editors, *Network & System Support for Games (NetGames 2006)*, pages 1–12. ACM Press, 2006. 32

[Vik05] Knut-Helge Vik. Game state and event distribution using proxy technology and application layer multicast. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 1041–1042, New York, NY, USA, 2005. ACM. 33

[WBB02] Talmadge Wright, Eric Boria, and Paul Breidenbach. Creative player actions in fps online video games - playing counter-strike. *Game Studies - the international journal of computer game research*, 2, Dezember 2002. 42

[WCC+09] Chen-Chi Wu, Kuan-Ta Chen, Chih-Ming Chen, Polly Huang, and Chin-Laung Lei. On the challenge and design of transport protocols for mmorpgs. *Multimedia Tools and Applications*, 45(1-3):7–32, 2009. 31

[WDX+06] Dmitri Williams, Nicolas Ducheneaut, Li Xiong, Yuanyuan Zhang, Nick Yee, and Eric Nickell. From tree house to barracks: The social life of guilds in world of warcraft. *Games and Culture*, 1(4):338–361, October 2006. 38

[Wei93] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):74–84, 1993. 15

[WFG92] Roy Want, Veronica Falcao, and Jon Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10:91–102, 1992. 15

[WJH97] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 4(5):42–47, 1997. 13

[WSA+96] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. *The Parctab Ubiquitous Computing Experiment*, volume 353 of *The International Series in Engineering and Computer Science*, chapter 2, pages 45–101. The Kluwer International Series in Engineering and Computer Science, mobile computing edition, 1996. 13, 15

[YC02] Jianxin Jeff Yan and Hyun-Jin Choi. Security issues in online games. *The Electronic Library*, 20(2):125–133, 2002. 79

[Yee06] Nick Yee. The demographics, motivations and derived experiences of users of massively-multiuser online graphical environments. *PRESENCE: Teleoperators and Virtual Environments*, 15:309–329, 2006. 1, 30

[YMYI05] Shinya Yamamoto, Yoshihiro Murata, Keiichi Yasumoto, and Minoru Ito. A distributed event delivery method with load balancing for mmorpg. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–8, New York, NY, USA, 2005. ACM. 32

[YR05] Jeff Yan and Brian Randell. A systematic classification of cheating in online games. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–9, New York, NY, USA, 2005. ACM. 79

[ZA04] Sebastian Zander and Grenville Armitage. Empirically measuring the qos sensitivity of interactive online game players. In *Proceedings of the Australian Telecommunications Networks and Applications Conference*, pages 511–518, Bondi Beach, Sydney, Australia, 8.-10. December 2004. 33

Online Referenzen

[1] *German Lab*. http://www.german-lab.de/. – Stand: 10.07.2009 121

[2] Activision: *Enemy Territory: Quake Wars, Official PC Stats Site*. http://stats.enemyterritory.com. – Stand: 10.07.2009 81, 82

[3] ApacheSoftwareFoundation: *Apache Axis2*. http://ws.apache.org/axis2/. – Stand: 10.06.2009 103

[4] AppacheJakartaProject: *JMeter*. http://jakarta.apache.org/jmeter/. – Stand: 10.06.2009 121

[5] ArenaNet: *Guild Wars*. http://de.guildwars.com/. Version: 2007. – Stand: 01.07.2009 29, 110

[6] Bigpoint: *Gangs of Crime 1930*. http://www.mafia1930.de/?aid=61. – Stand: 17.04.2010 81

[7] BlizzardEntertainment: *The World of Warcraft Armory*. http://www.wowarmory.com/. – Stand: 10.06.2009 81, 82

[8] BlizzardEntertainment: *Starcraft*. http://www.blizzard.com/us/starcraft/. Version: 1998. – Stand: 01.07.2009

[9] BlizzardEntertainment: *Diablo II*. http://www.blizzard.com/us/diablo2/. Version: 2000. – Stand: 01.07.2009

[10] BlizzardEntertainment: *Warcraft III: Reign of Chaos*. http://www.blizzard.com/us/war3/. Version: 2002. – Stand: 01.07.2009 29

[11] BlizzardEntertainment: *World of Warcraft*. http://www.wow-europe.com/de/index.xml. Version: 2007. – Stand: 01.07.2009 5, 30, 41, 104

[12] BlizzardEntertainment: *WORLD OF WARCRAFT zählt jetzt mehr als 11,5 Millionen Abonnenten weltweit*. http://eu.blizzard.com/de/press/081223.html. Version: Dezember 2008. – Stand: 10.06.2009 30, 42

[13] Bray, Tim ; Paoli, Jean ; Sperberg-McQueen, C.M. ; Maler, Eve ; Yergeau, Francois: Extensible Markup Language (XML) 1.0 (Fifth Edition) / W3C. Version: November 2008. http://www.w3.org/TR/2008/REC-xml-20081126/. 2008. – W3C Recommendation. – Stand: 10.06.2009 65

[14] Brazie, Alexander ; Thott ; AnduinLothar: *Chronos*. http://www.cosmosui.org/addons.php?info=Chronos. Version: 2008. – Stand: 10.06.2009 106

[15] Buckley, Tim: *CTRL+ALT+DEL*. http://www.ctrlaltdel-online.com/index.php. – Stand: 10.06.2009 41

[16] CCP: *EVE Online*. http://www.eveonline.com/. Version: 1997. – Stand: 01.07.2009

[17] ComputecMediaAG: *buffed.de - Das Portal für Online Spiele*. http://www.buffed.de/. – Stand: 10.06.2009 46, 83

[18] Curse, Inc.: *Curse.com - WoW Addons WAR Addons Gamer Blogs Forums*. http://www.curse.com/. – Stand: 10.07.2009 42

[19] ElectronicArts: *Battlefield*. Series. http://www.battlefield.ea.com/battlefield/bf/. – Stand: 01.07.2009

[20] ElectronicArts: *EA Video Games - Electronic Arts*. http://www.ea.com/. – Stand: 10.07.2009 84

[21] ElectronicArts: *Need for Speed*. Series. http://www.needforspeed.com/. – Stand: 01.07.2009

[22] ElectronicArts: *Rupture*. http://www.rupture.com/. – Stand: 10.07.2009 83

[23] ElectronicSportsLeague: *ESL Wire*. http://www.esl.eu/de/wire/. – Stand: 10.09.2008 83

[24] EpicGames: *Unreal Tournament*. Series. http://www.unrealtournament2004.com/utgoty/. – Stand: 01.07.2009

[25] EvenBalance: *PunkBuster*. http://www.evenbalance.com/. – Stand: 24.07.2009 79

[26] Fielding, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures*, UNIVERSITY OF CALIFORNIA, IRVINE, Diss., 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. – Stand: 15.07.2009 103

[27] FlagshipStudios: *Hellgate London*. http://www.hellgatelondon.de/. – Stand: 17.04.2010 29

[28] Forbes.com: *The Most Revolutionary Videogames Of All Time*. http://www.forbes.com/2009/12/07/video-games-wii-technology-revolutionaries-09-games.htmlhttp://www.forbes.com/2009/12/07/video-games-wii-technology-revolutionaries-09-games_slide_11.html. – Stand: 15.07.2009 106

[29] GarageGames.com: *OpenTNL - Torque Network Library Reference*. http://opentnl.sourceforge.net/doxydocs/. – Stand: 15.07.2009 102

[30] Ierusalimschy, R. ; Figueiredo, L. H. ; Celes, W.: *Lua 5.1 Reference Manual*. Lua.org, 2006 http://www.lua.org/manual/5.1/. – Stand: 15.07.2009 42, 105

[31] IroncladGames ; KalypsoMedia: *Sins of a Solar Empire*. http://www.sinsofasolarempire.com/. Version: 2008. – Stand: 01.04.2010

[32] JenkinsSoftware, LLC: *RakNet - Multiplayer game network engine*. http://www.jenkinssoftware.com/. – Stand: 15.07.2009 102

[33] lua.org: *Lua About*. http://www.lua.org/about.html. – Stand: 09.12.09 105

[34] lua.org: *Lua Uses*. http://lua-users.org/wiki/LuaUses. – Stand: 09.12.09 105

[35] Machinima, Inc.: *Machinima.com*. http://www.machinima.com/. – Stand: 15.07.2009 41

[36] Maxis ; ElectronicArts: *SPORE*. www.spore.com/. Version: 2009. – Stand: 15.07.2009 42

[37] Mücke, Christian: *WoW Interface Design oder User Generated Content der üblen Art*. http://www.coldheat.de/archiv/2008/05/wow-interface-design-oder-user-generated-content-der-ueblen-art. http://www.coldheat.de/archiv/2008/05/wow-interface-design-oder-user-generated-content-der-ueblen-art. Version: Mai 2008. – Stand: 24.07.2009 10

[38] MediaMolecule ; SonyComputerEntertainment: *LittleBIGPlanet*. http://www.littlebigplanet.com/. Version: 2008. – Stand: 15.07.2009 42

[39] Microsoft: *Age of Empires*. Series. http://www.microsoft.com/games/empires/. – Stand: 01.07.2009

[40] Mills, David L.: *Network Time Protocol*. RFC 1305. http://tools.ietf.org/html/rfc1305. Version: March 1992. – Stand: 15.07.2009 110

[41] mTw, SportsLimited: *mTw - 360 degrees gaming*. http://www.mymtw.com/de/. – Stand: 10.07.2009 40

[42] MumbleTeam: *Mumble*. http://mumble.sourceforge.net/. – Stand: 15.07.2009 109

[43] MythicEntertainment ; GOA ; ElectronicArts: *Warhammer Online : Age of Reckoning*. http://www.warhammeronline.com/. Version: 2008. – Stand: 15.07.2009 39

[44] Nadeo: *Trackmania*. Series. http://www.trackmania.com/. – Stand: 01.04.2010

[45] NextIdea, GmbH ; 2bMedia, GmbH: *Space Pioneers*. http://gde.sp.looki.de/glogin.shtml. Version: 2004. – Stand: 01.07.2009

[46] ObjectManagementGroup, Inc.: *CORBA*. http://www.corba.org/. – Stand: 24.07.2009 22

[47] OriginSystems ; ElectronicArts: *Ultima Online*. http://www.uoherald.com/news/. Version: 1997. – Stand: 24.07.2009 104

[48] Pannier, Oskar ; Clifford, Marvin: *Shakes & Fidget*. http://www.shakes-and-fidget.com/. – Stand: 24.07.2009 41

[49] Rüppel, Jörg: *Zoidcom Network Library*. http://www.zoidcom.com/. – Stand: 24.07.2009 102

[50] Salzman, Lee: *ENet*. http://enet.bespin.org. – Stand: 24.07.2009 102

[51] Scan ; Zedd: *Dark Galaxy*. http://www.darkgalaxy.com. – Stand: 13.02.2008

[52] Sohmer, Ryan ; DeSouza, Lar: *Looking for Group*. http://www.lfgcomic.com/. – Stand: 24.07.2009 41

[53] SonyOnlineEntertainment: *Star Wars Galaxies*. http://starwarsgalaxies.station.sony.com/players/index.vm. Version: 2003. – Stand: 01.07.2009

[54] Stevinho: *Allimania*. http://www.wowszene.de/download.php?list.7. – Stand: 24.07.2009 41

[55] SunMicrosystems: *Java Message Service (JMS)*. http://java.sun.com/products/jms/. – Stand: 24.07.2009 102

[56] SunMicrosystems ; GlassfishCommunity: *Java API for XML- Web Services*. https://jax-ws.dev.java.net/. – Stand: 24.07.2009 103

[57] TeamSpeakSystems, GmbH: *TeamSpeak*. http://www.teamspeak.com/. – Stand: 04.06.2009 40, 46

[58] Thott: *THOTTBOT: World of Warcraft Database*. http://thottbot.com/. – Stand: 24.07.2009 40

[59] TodecMedia: *SchoolWars*. http://www.schoolwars.de/. – Stand: 17.04.2010 81

[60] TravianerGames, GmbH: *Travianer*. http://www.travianer.de/. – Stand: 01.04.2010

[61] TurtleEntertainment: *ESL: Electronic Sports League - The eSports Gaming League*. www.esl.eu/. – Stand: 10.07.2009 83

[62] Valve: *Steam*. http://store.steampowered.com/. – Stand: 24.07.2009 83

[63] VALVE: *Valve Corporation*. http://www.valvesoftware.com/. – Stand: 24.07.2009 42, 83

[64] VALVE: *Half-Life*. http://www.valvesoftware.com/games.html. Version: 1998. – Stand:24.07.2009 41, 42

[65] VALVE: *Counter-Strike*. http://www.valvesoftware.com/games.html. Version: 2000. – Stand: 24.07.2009 42

[66] VALVE: *Counter-Strike: Source*. http://www.valvesoftware.com/games.html. Version: 2004. – Stand: 24.07.2009 42

[67] VIACOM: *Xfire - Gaming Simplified*. http://www.xfire.com/. – Stand: 10.07.2009 83

[68] VOIG, Inc.: *MMOGData: Charts*. http://mmogdata.voig.com/. – Stand: 24.07.2009 30

[69] W3C: *DTD - Document Type Declaration*. http://www.w3.org/TR/REC-xml/#dt-doctype. – Stand: 10.07.2009 65

[70] W3C: *Extensible Markup Language (XML)*. http://www.w3.org/XML/. – Stand: 10.07.2009 64

[71] W3C: *Resource Description Framework (RDF)*. http://www.w3.org/RDF/. – Stand: 10.07.2009 64

[72] W3C: *World Wide Web Consortium*. http://www.w3.org/. – Stand: 10.07.2009 65

[73] W3C: *XML Schema*. http://www.w3.org/XML/Schema. – Stand: 10.07.2009 65

[74] W3C: *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. http://www.w3.org/TR/soap12-part1. Version: 2007. – Editors: Martin Gudgin and Marc Hadley and Noah Mendelsohn and Jean-Jacques Moreau and Henrik Frystyk Nielsen and Anish Karmarkar and Yves Lafon - Stand: 10.07.2009 103

[75] W3SCHOOLS: *Introduction to XML Schema*. http://www.w3schools.com/schema/schema_intro.asp. – Stand: 09.12.09 65

[76] WEBGUIDEZENTERTAINMENT: *gamona.de*. http://www.gamona.de/. – Stand: 10.07.2009 40

[77] WIKIA: *Guildwiki: a wiki guide for ArenaNet's Competitive Online RPG, Guild Wars*. http://guildwars.wikia.com/wiki/Main_Page. – Stand: 17.12.08 40

[78] WIZARDSOFTHECOAST: *Dungeons & Dragons*. http://www.wizards.com/default.asp?x=dnd/welcome. Version: 1974. – Stand: 23.06.09 26

[79] WOODCOCK, Bruce: *An Analysis of MMOG Subscription Growth*. http://www.mmogchart.com/charts/. – Stand: 01.07.2009 30, 31

[80] XCHAR: *Xchar, die WoW Community - WoW meets Real Life - Spieler, Chars & Gilden aus World of Warcraft*. http://www.xchar.de/. – Stand: 10.07.2009 83

Verzeichnis aller Abkürzungen

afk	away from keyboard - nicht an der Tastatur
API	Application Programming Interface
CoBrA	Context Broker Architecture
DD	Damage Dealer
dSK-Dienst	dynamischer SprachKommunikationsdienst
DTD	Document Type Definition
e-sport	Electronic Sport
ESL	Electronic Sports League
EULA	End User License Agreement
FOV	Field of View
FPS	First Person Shooter
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HUD	Heads-Up Display
IIO	IngameInformationObject
IM	Instant Messanger
JMS	Java Messaging Service
LBA-Dienst	LokationsBasierter Annotationsdienst
MMOG	Massively Multiplayer Online Game
MMORPG	Massively Multiplayer Online RolePlaying Game
Mod	Modification - Modifikation
MOG	Multiplayer Online Game
MUD	Multi-User-Dungeon
NPC	Non Player Character - Nichtspieler Charakter
NVE	Networked Virtual Environment
P2P	peer-to-peer
PvE	Player versus Environment
PvP	Player versus Player

RDF	Resource Description Framework
REST	Representional State Transfer
RPC	Remote Procedure Call
RPG	RolePlaying Game - Rollenspiel
RTS	RealTime Strategy Game - Echtzeitstrategiespiel
RvR	Realm versus Realm
SGML	Standard Generalized Markup Language
SMOG	Session Multiplayer Online Game
SOCAM	Service-Oriented Context-Aware Middleware
SUI	Service User Interface
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VCBS	Virtual Context Based Services - virtuell kontextbasierte Dienste
VP	VirtualParameter - virtuelle Parameter
W3C	World Wide Web Consortium
WCG	World Cyber Games
WoW	World of Warcaft
xgdl	eXtensible Game Description Language
XML	Extensible Markup Language
XSD	XML-Schema-Definition

Anhang

A Weitere Details zu Multiplayer Online Games, Kontextbeschreibung und Interaktionsmodellierung

A.1 Weitere Details zu Multiplayer Online Games

Es lassen sich durchaus Ähnlichkeiten zwischen den Eigenschaften virtueller Welten und realer Ökonomien feststellen. Gerade die Symbiose aus fiktiven und realen Ansätzen ermöglicht es dem Spieler, sich ohne lange Eingewöhnungsphase in einem Spiel zurechtzufinden und regelrecht darin einzutauchen. Um den wirtschaftlichen Erfolg eines MMORPGs sicherzustellen, gilt es, eine gesunde Mischung aus realistischen Funktionsweisen und fiktiven, unterhaltenden Elementen zu finden. Von fast ebenso großer Bedeutung ist ein ausgeklügeltes Wirtschaftssystem mit der Aufgabe eine große Anzahl unterschiedlicher Spielertypen für eine unbegrenzte Zeit zu binden. Diesbezüglich ist eine kontinuierliche Erweiterung der Spielinhalte (*Content*) von Nöten, um den Spielspaß zu erhalten. Die Spielwelt mit ihren Eigenschaften (*Features*), insbesondere die sozialen Interaktionsmöglichkeiten, stellen den Rahmen eines MMORPGs dar.

Features

Features sind die Eigenschaften eines Spiels. Diese umfassen beispielsweise das Vorhandensein von Tages- und Nachtzeiten, die verwendeten Ansichten oder die verschiedenen Möglichkeiten, die ein Spiel bietet, wie beispielsweise das Verwenden von Fahrzeugen. Ein Spiel definiert sich über seine Features. Viele Features findet man in fast allen Spielkonzepten wieder. Den Unterschied machen die Vielfältigkeit, die Innovation und die Handhabbarkeit der Features aus. Unternehmen werben mit neu entwickelten Features eines Spiels, weil sie für Fortschritt stehen. Ein MOG hat besondere Anforderungen an seine Features, denn diese müssen für Ausgeglichenheit sorgen. Da selten die Wirkung aller Features zum Release überschaubar ist, werden diese mit Patches und Updates nach und nach angepasst, um das Spielgefüge im Gleichgewicht zu halten (Balancing). Typische Beispiel für Features von Spielen sind: Mehrspielermodus (Multiplayer), Online spielbar, 3D-Grafik, kooperative Spielweise, persistente Spielwelt, künstliche Intelligenz, Physik-Engine, Wirtschaftssystem oder ein anpassbares User-Interface. Die Liste ließe sich lange fortsetzten.

Content - Spielinhalt

Content bezeichnet den Inhalt der Spielwelt. Er umfasst den gestalterischen Aspekt eines Spiels und sorgt für Vielfalt und Abwechslung im Spielverlauf. Das beinhaltet große Gebiete oder Maps, schöne Landschaften, viele Aufgaben (Quests) und Gegenstände. Der Unterschied zwischen Content und Features liegt in der Ausgangsposition. Features stellen die Handlungsweisen und -möglichkeiten bereit, bilden somit das Rahmenwerk in dem sich das Spiel befindet. Content soll das Spiel füllen und sorgt für die Inhalte. Features beschreiben die qualitative Dimension des Spiels. Content im engeren Sinne ist eher ein quantitatives Merkmal, also wie viele Quests gibt es im Spiel, wie viele Ausbaustufen können durchlaufen werden, wie viele unterschiedliche Einheiten können produziert werden. Um das Interesse der Community über einen langen Zeitraum hinweg zu gewinnen, müssen immer wieder neue Spielinhalte eingebracht werden, die dem Spiel neue Impulse und den Spielern eine Beschäftigung geben.

Developer und Publisher

Die Spieleindustrie setzt sich im Prinzip aus zwei Playern zusammen. Auf der einen Seite gibt es die Developer (Entwickler), sie produzieren in der Regel das Spiel von der Idee bis zur Marktreife. Da die Entwicklung viel Geld kostet, treten die Entwickler frühzeitig an Publisher heran. Publisher sind ver-

gleichbar mit Verlagen, die dafür sorgen, dass ein Spiel veröffentlicht und vermarktet wird. In einem Vertrag verpflichtet sich der Entwickler das Spiel zu entwickeln und der Publisher dies zu finanzieren und erhält im Gegenzug die Vermarktungsrechte. Dieses Verhältnis ist in der Regel solange problemlos, bis eine Seite die Zusammenarbeit beendet. Dann entstehen häufig Rechtsstreitigkeiten über Namens- und Nutzungsrechte. Beide Seiten haben ein berechtigtes Interesse daran, Technologie und Namen weiter zu verwenden. Der Titel eines erfolgreichen Spiels stellt ein großes Kapital dar, denn Spiele, die unter diesem Namen veröffentlicht werden, lassen sich leicht am Markt absetzen. Ebenso interessant kann eine innovative Engine sein, die einen technologischen Wettbewerbsvorteil gegenüber der Konkurrenz bedeutet. Durch die finanziell schwächere Position der Entwickler müssen sich diese häufig auf die Konditionen der Publisher einlassen, sodass am Ende der Kooperation die Nutzungsrechte oft beim Publisher verbleiben. Dieser kann dann ein anderes Entwicklerteam beauftragen, das auf Grundlage der Nutzungsrechte für den Publisher weiterarbeitet.

A.2 XML-Schema der xgdl

XML-Schema der Basis-xgdl

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- this is the basic schema corresponding to the generic game description
    XGDL -->

<!-- definition of types -->
<xs:complexType name="parameterLable">
        <xs:sequence>
                <xs:element name="vpCategory" type="category"/>
                <xs:element name="vpSet" type="set"/>
                <xs:element name="vpName" type="virtualParam"/>
        </xs:sequence>
</xs:complexType>
<xs:simpleType name="category">
        <xs:restriction base="xs:string">
                <xs:enumeration value="general"/>
                <xs:enumeration value="location"/>
                <xs:enumeration value="character"/>
                <xs:enumeration value="status"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="set">
        <xs:restriction base="xs:string">
                <!-- general sets -->
                <xs:enumeration value="Player␣info"/>
                <xs:enumeration value="Game␣time"/>
                <xs:enumeration value="Server␣info"/>
                <!-- location sets -->
                <xs:enumeration value="Position"/>
                <!-- character sets -->
                <xs:enumeration value="Character␣info"/>
                <xs:enumeration value="Equipment"/>
```

```
                <!-- status sets -->
                <xs:enumeration value="State"/>
                <xs:enumeration value="Group"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="virtualParam">
        <xs:restriction base="xs:string">
        <!-- general parameters -->
                <!-- general - player info parameters -->
                <xs:enumeration value="player_name"/>
                <!-- general - game time parameters -->
                <xs:enumeration value="overall_time"/>
                <xs:enumeration value="session_time"/>
                <!-- general - server info parameters -->
                <xs:enumeration value="server_name"/>
                <xs:enumeration value="server_type"/>
        <!-- location parameters -->
                <!-- location - position parameters -->
                <xs:enumeration value="xcoordinate"/>
                <xs:enumeration value="ycoordinate"/>
                <xs:enumeration value="zcoordinate"/>
        <!-- character parameters -->
                <!-- character - character info parameters -->
                <xs:enumeration value="character_name"/>
                <xs:enumeration value="character_type"/>
                <!-- character - equipment parameters -->
                <xs:enumeration value="equipped"/>
        <!-- status parameters -->
                <!-- status - state parameters -->
                <xs:enumeration value="action"/>
                <!-- status - group parameters -->
                <xs:enumeration value="grouped"/>
                <xs:enumeration value="group_size"/>
        </xs:restriction>
</xs:simpleType>


<xs:element name="sValue" type="xs:string"/>
<xs:element name="lValue" type="xs:string"/>

<xs:group name="range">
        <xs:sequence>
                <xs:element name="from" type="xs:integer"/>
                <xs:element name="to" type="xs:integer"/>
        </xs:sequence>
</xs:group>

<xs:complexType name="valueType">
        <xs:sequence>
                <xs:element ref="sValue" minOccurs="0"/>
```

```
                <xs:element ref="lValue" minOccurs="0" maxOccurs="unbounded
                    "/>
                <xs:group ref="range" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="entryType">
        <xs:sequence>
                <xs:element name="key" type="parameterLable"/>
                <xs:element name="value" type="valueType"/>
        </xs:sequence>
</xs:complexType>


</xs:schema>
```

XML-Schema der MMOG-xgdl

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">


<!-- definition of types -->
<xs:complexType name="parameterLable">
        <xs:sequence>
                <xs:element name="vpCategory" type="category"/>
                <xs:element name="vpSet" type="set"/>
                <xs:element name="vpName" type="virtualParam"/>
        </xs:sequence>
</xs:complexType>
<xs:simpleType name="category">
        <xs:restriction base="xs:string">
                <xs:enumeration value="general"/>
                <xs:enumeration value="location"/>
                <xs:enumeration value="character"/>
                <xs:enumeration value="status"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="set">
        <xs:restriction base="xs:string">
                <!-- general sets -->
                <xs:enumeration value="Game␣info"/>
                <xs:enumeration value="Player␣info"/>
                <xs:enumeration value="Game␣time"/>
                <xs:enumeration value="Server␣info"/>
                <xs:enumeration value="Buddy␣list"/>
                <!-- location sets -->
                <xs:enumeration value="Position"/>
                <xs:enumeration value="Area␣information"/>
                <!-- character sets -->
                <xs:enumeration value="Character␣info"/>
                <xs:enumeration value="Equipment"/>
```

```
                <xs:enumeration value="Skills"/>
                <!-- status sets -->
                <xs:enumeration value="State"/>
                <xs:enumeration value="Group"/>
                <xs:enumeration value="Encounters"/>
                <xs:enumeration value="Guild"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="virtualParam">
        <xs:restriction base="xs:string">
        <!-- general parameters -->
                <!-- general - game info parameters -->
                <xs:enumeration value="game_name"/>
                <!-- general - player info parameters -->
                <xs:enumeration value="player_name"/>
                <!-- general - game time parameters -->
                <xs:enumeration value="overall_time"/>
                <xs:enumeration value="session_time"/>
                <!-- general - server info parameters -->
                <xs:enumeration value="server_name"/>
                <xs:enumeration value="server_type"/>
                <!-- general - buddy list parameters -->
                <xs:enumeration value="buddies"/>
        <!-- location parameters -->
                <!-- location - position parameters -->
                <xs:enumeration value="x_coordinate"/>
                <xs:enumeration value="y_coordinate"/>
                <xs:enumeration value="z_coordinate"/>
                <!-- location - area information parameters -->
                <xs:enumeration value="area_name"/>
                <xs:enumeration value="area_type"/>
                <xs:enumeration value="area_level"/>
        <!-- character parameters -->
                <!-- character - character info parameters -->
                <xs:enumeration value="character_name"/>
                <xs:enumeration value="character_type"/>
                <xs:enumeration value="character_class"/>
                <xs:enumeration value="character_race"/>
                <xs:enumeration value="character_level"/>
                <!-- character - equipment parameters -->
                <xs:enumeration value="equipped"/>
                <xs:enumeration value="equipped_amor"/>
                <xs:enumeration value="equipped_weapon"/>
                <xs:enumeration value="bag"/>
                <xs:enumeration value="bank"/>
                <!-- character - skills parameters -->
                <xs:enumeration value="active_skills"/>
                <xs:enumeration value="crafting"/>
        <!-- status parameters -->
                <!-- status - state parameters -->
```

```
                <xs:enumeration value="action"/>
                <xs:enumeration value="health"/>
                <xs:enumeration value="power"/>
                <!-- status - group parameters -->
                <xs:enumeration value="grouped"/>
                <xs:enumeration value="group_size"/>
                <!-- status - encounters parameters -->
                <xs:enumeration value="pvp_pve"/>
                <!-- status - guild parameters -->
                <xs:enumeration value="guilded"/>
                <xs:enumeration value="guild_name"/>
                <xs:enumeration value="guild_size"/>
                <xs:enumeration value="guild_rank"/>
        </xs:restriction>
</xs:simpleType>

<xs:element name="sValue" type="xs:string"/>
<xs:element name="lValue" type="xs:string"/>

<xs:group name="range">
        <xs:sequence>
                <xs:element name="from" type="xs:integer"/>
                <xs:element name="to" type="xs:integer"/>
        </xs:sequence>
</xs:group>

<xs:complexType name="valueType">
        <xs:sequence>
                <xs:element ref="sValue" minOccurs="0"/>
                <xs:element ref="lValue" minOccurs="0" maxOccurs="unbounded
                    "/>
                <xs:group ref="range" minOccurs="0"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="entryType">
        <xs:sequence>
                <xs:element name="key" type="parameterLable"/>
                <xs:element name="value" type="valueType"/>
        </xs:sequence>
</xs:complexType>



</xs:schema>
```

A.3 Interaktionsmodellierung

Ziel der Interaktionsmodellierung ist, eine Formalisierung der Kooperation vorzunehmen. Die Kooperation besteht aus dem kooerdinierten Handeln mehrerer Spieler mit einem gemeinsamen Ziel. Eine Formalisierung ermöglicht eine Modellierung und Analyse von Kooperationssituationen durch die Bereitstellung

eines konkreten spielunabhängigen Vokabulars. Darüber hinaus wird dadurch auch eine Unterstützung der Planung von Kooperationssituationen ermöglicht. Im Vordergrund steht dabei die Kooperation einer Spielergruppe in einer aktiven Spielphase. Für eine umfassendere Betrachtung müssen auch die vorbereiteten Aktionen, wie Gruppenfindung oder Planung (Phase 1) und die Nachbereitung, wie Besprechung und Beuteverteilung (Phase 3), berücksichtigt werden (siehe Abbildung A.1).

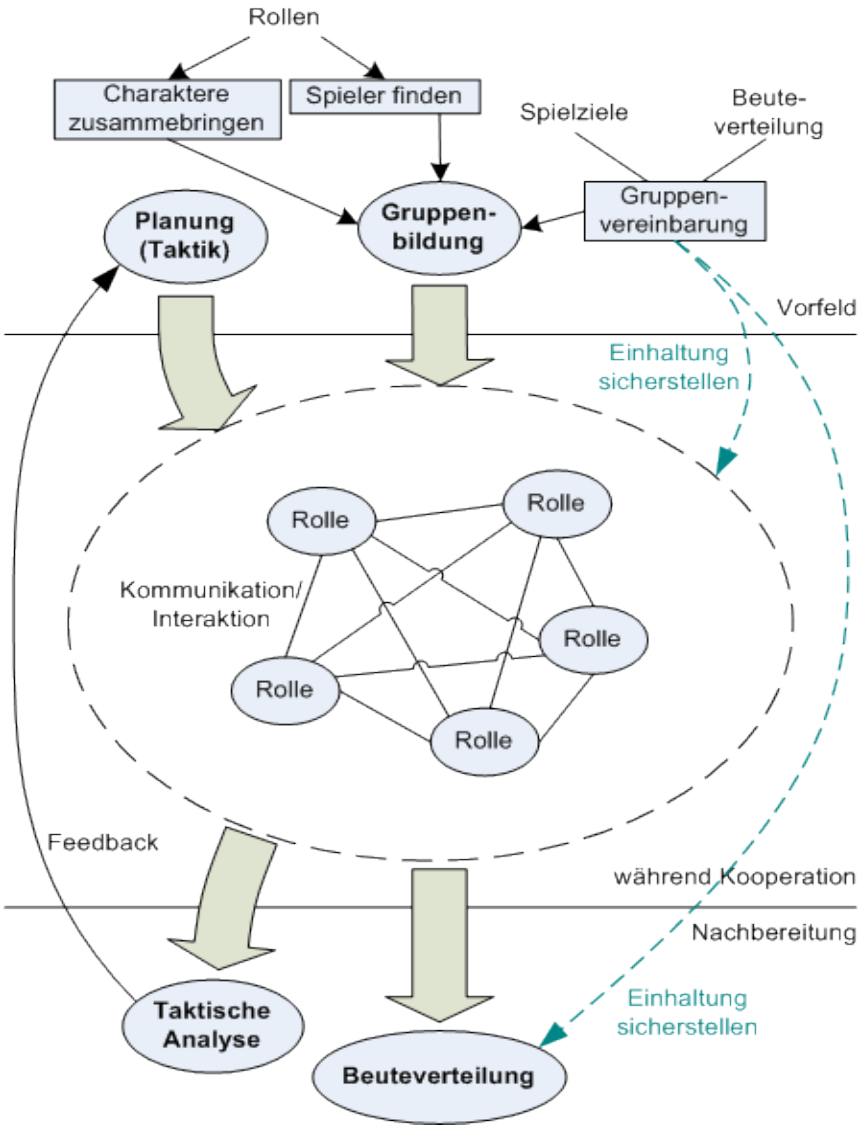

Abbildung A.1: Komponenten des Interaktionsmodells

Die Taktik, mit der eine Gruppe ein Ziel verfolgt, bildet den Ausgangspunkt der Modellierung von Phase 2. Dabei kann ein Ziel sehr komplex sein und aus mehreren Unteraufgaben bestehen, die gleichzeitig oder nacheinander erfüllt werden müssen. Das koordinierte gemeinsame Vorgehen zum Erreichen der einzelnen Teilziele und des Gesamtziels bildet den Aktionsplan der Kooperationssituation. Dieser definiert, welcher Spieler unter welchen Umständen welche Aktionen durchführen soll. Die einzelnen Aspekte eines Aktionsplans sind die *Aktionen*, die ausgeführt werden und *Auslöser*, die die Ausführung von Aktionen triggern. **Aktionen** sind die Handlungen der an der Kooperationssitation beteiligten Spieler. Diese können unterteilt werden in Aktionen auf Charakterebene, die der Spieler über seinen virtuellen Charakter ausführt (wie einem Gegner Schaden zufügen oder ein Gruppenmitglied heilen) und Aktionen auf Spielerebene, die er unabhängig von seinem Charakter ausführt (wie eine Text-Chat Nachricht schreiben). Die Aktionen müssen zur vollständigen Beschreibung mit ein oder zwei Objekten parameterisiert

werden, im Normalfall wird neben dem Subjekt der Aktion (dem Spieler bzw. seinem Charakter) auch das Objekt der Aktion festgelegt. **Auslöser** sind die zeitlichen und kausalen Aspekte das Ablaufplans. Dafür wird ein ereignisbasiertes System benötigt, da eine dynamische Reaktion auf Ereignisse innerhalb der Spielwelt abgebildet werden muss. Ein Auslöser definiert, wann oder unter welchen Bedingungen ein Teil des Plans (Teilplan) durchgeführt wird. Neben den Auslösern kann ein Teilplan noch zusätzliche Nebenbedingungen aufweisen. Es können vier Arten von Auslösern unterschieden werden: Feste Zeitpunkte, Kommunikationsereignisse, beobachtete Ereignisse im Spiel und Änderungen des Spielzustands (eigener Charakter).

Die graphische Darstellung der Modellierung der Interaktionsabläufe orientiert sich an existierenden Modell-Graphiksprachen, wobei UML-Sequenzdiagramme als Referenz verwendet werden. UML-Sequenzdiagramme werden im Softwaredesign verwendet, um sich wechselseitig steuernde Abläufe zwischen einzelnen Programmobjekten darzustellen. In Sequenzdiagrammen handelt es sich bei diesen Abläufen um Methoden der verschiedenen Objekte, die über Aufrufe initiiert werden.

Ersetzt man die Objekte durch Rollen, die Aufrufe durch Auslöser und die Methoden durch Aktionen, so erhält man eine geeignete Basis für die graphische Darstellung der kooperativen Abläufe. Außerdem unterscheidet sich die grundlegende Semantik nicht vom originalen UML-Sequenzdiagramm, welches in der Informatik weit verbreitet ist. Dadurch ist das Kooperationsdiagramm für Leser mit UML-Vorkenntnissen leicht verständlich. So werden z.B. Nebenbedingungen der Auslöser in der UML-üblichen Form in eckigen Klammern annotiert. Eine grundlegende Modifikation wurde jedoch vorgenommen: Eine Drehung der Zeitachsen von der Vertikalen in die Horizontale verschafft eine bessere Möglichkeit zur Beschriftung der Aktionen und bringt keine signifikanten Nachteile mit sich. An dieser Stelle zeigen sich auch gewisse Ähnlichkeiten zur häufig verwendeten Darstellungsform für Kommunikations- oder Bus-Protokolle.

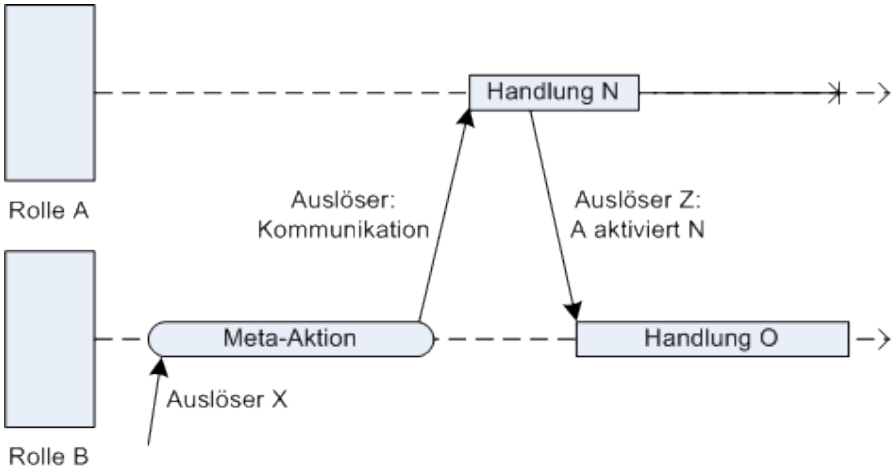

Abbildung A.2: Komponenten des Interaktionsmodells

Abbildung A.2 zeigt ein beispielhaftes Diagramm, in dem sämtliche relevanten Komponenten Verwendung finden. Es wird graphisch differenziert zwischen Aktionen auf Spielerebene (Meta-Aktionen), die durch Rechtecke mit abgerundeten Ecken dargestellt werden und solchen auf Charakterebene, dargestellt durch Rechtecke auf der Lebenslinie. Aktionen können außerdem einmalig durchgeführt werden, was der bereits vorgestellten Darstellungsform entspricht, oder als "anhaltende Aktionen" dargestellt mit einem fortlaufenden Pfeil entlang der Lebenslinie (siehe Handlung N). Bis zu einem Auslöser, definiert am Ende dieses Pfeils, wird diese Aktion immer wieder ausgeführt.

Im Folgenden finden sich Beispiele für die Modellierung typischer Kooperationssituationen in MOGs. Diese Umfassen die eigentliche Kooperation wie den Handel von virtuellen Gegenständen (Abbildung A.3), den Handel von Dienstleistungen am Beispiel einer Gegenstandsverbesserung (Verzauberung), der Ausrüstung des Interessenten durch den Anbieter (Abbildung A.4) und verschiedenen Kampfsituationen

(Abbildungen A.5, A.6, A.7 und A.8). Eine kombinierte Darstellung der Einzelaktionen findet sich in Abblindung 4.7 in Abschnitt 4.1.4.

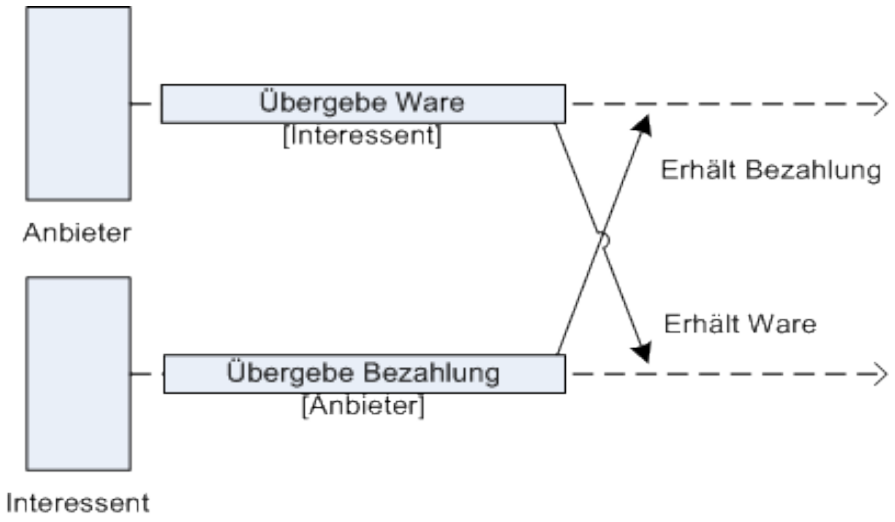

Abbildung A.3: Interaktionsmodell: Handel von virtuellen Gegenständen

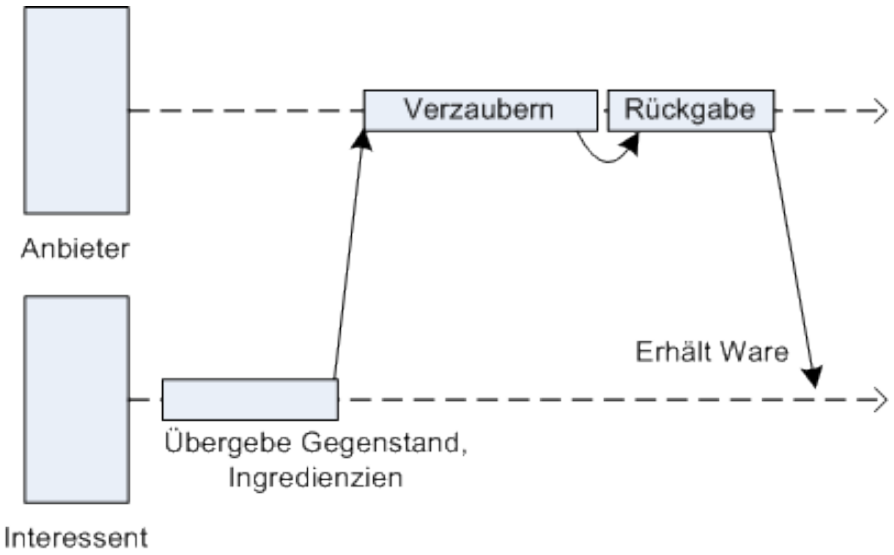

Abbildung A.4: Interaktionsmodell: Handel von Dienstleistungen

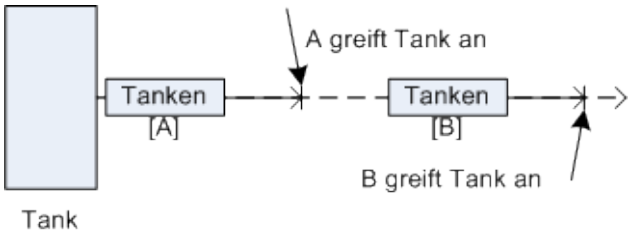

Abbildung A.5: Interaktionsmodell: Begegnung zwischen einem Tank und zwei Gegnern

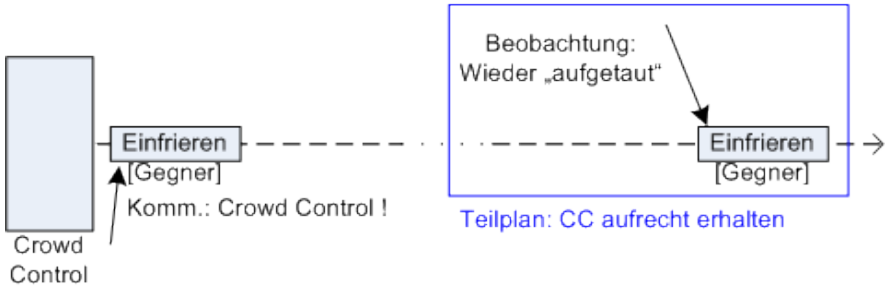

Abbildung A.6: Interaktionsmodell: "Crowd Control" durch die Fähigkeit "Einfrieren"

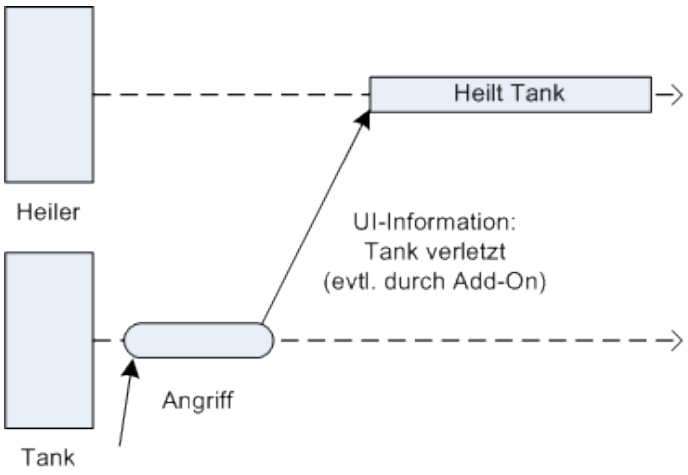

Abbildung A.7: Interaktionsmodell: Kooperationsgraph Heiler und Tank

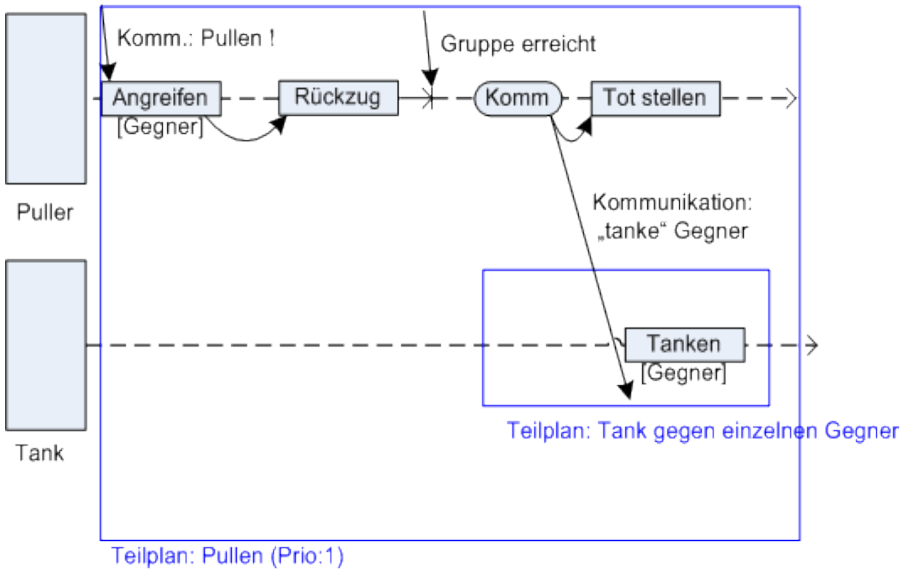

Abbildung A.8: Interaktionsmodell: Anlocken von Gegnern ("Pullen")

B Liste der eigenen Publikationen

B.1 Veröffentlichungen als Erstautorin

1. Sonja Bergsträßer, Tomas Hildebrandt, Christoph Rensing, Ralf Steinmetz: Virtual context based services for multiplayer online games to facilitate community participation. In: Multimedia Tools and Applications, May 2009. ISSN 1380-7501 (Print) 1573-7721 (Online). http://www.springerlink.com/content/78t28j1672k72021/?p=23c507d05e51450f80e3a325296edf01?=15.

2. Sonja Bergsträßer, Tomas Hildebrandt, Christoph Rensing, Ralf Steinmetz: A Middleware for the Controlled Information Exchange between Online Games and Internet Applications. In: Klaus David, Kurt Geihs: 16. ITG/GI - Fachtagung Kommunikation in Verteilten Systemen 2009 - KiVS 2009, Kassel, Germany, Springer , March 2009. ISBN 978-3-540-92665-8. http://www.springerlink.com/content/78t28j1672k72021/.

3. Sonja Bergsträßer, Tomas Hildebrandt, Lasse Lehmann, Christoph Rensing, Ralf Steinmetz: Virtual Context Based Services for Support of Interaction in Virtual Worlds. In: Grenville Armitage: Netgames 2007, 6th Annual Workshop on Network and Systems Support for Games, Melbourne, Australia, p. 111–116, September 2007. ISBN 978-0-9804460-0-5.

4. Sonja Bergsträßer, Christoph Rensing, Birgit Zimmermann, Ralf Steinmetz: Building a Representation of Learning Resources to Support their Re-Purposing. In: Proceedings of World Conference on Educational Multimedia, Hypermedia, June 2007.

5. Sonja Bergsträßer, Andreas Faatz, Christoph Rensing, Ralf Steinmetz: A Semantic Content Representation Supporting Re-Purposing of Learning Resources. In: Klaus Tochtermann, Hermann Maurer (Edits.): Proceedings of I-KNOW '06, p. 287–295, September 2006. ISBN 0948-695x.

6. Sonja Bergsträßer, Birgit Zimmermann, Marek Meyer, Christoph Rensing, Andreas Faatz, Tomas Hildebrandt, Ralf Steinmetz: Re-Purposing: Motivation, Related Work, and Building Blocks. no. KOM-TR-2005-03, December 2005.

B.2 Mitautorenschaft

7. Christian Gottron, André König, Matthias Hollick, Sonja Bergsträßer, Tomas Hildebrandt, Ralf Steinmetz: Quality of Experience of Voice Communication in Large-Scale Mobile Ad Hoc Networks. In: Proceedings of the second IFIP Wireless Days 2009, December 2009.

8. Tomas Hildebrandt, Sonja Bergsträßer, Christoph Rensing, Ralf Steinmetz: Capturing and Storing Profile Information for Gamers Playing Multiplayer Online Games. In: Maha Abdallah: NetGames 2009, 8th Annual Workshop on Network and Systems Support for Games, November 2009. ISBN 978-1-4244-5604-8.

9. Tomas Hildebrandt, Sonja Bergsträßer, Christoph Rensing, Ralf Steinmetz: Dynamic Voice Communication Support for Multiplayer Online Games. In: Marc Claypool: Netgames 2008, 7th Annual Workshop on Network and Systems Support for Games, p. 105–111, ACM, October 2008. ISBN 978-1-60558-132-3.

http://portal.acm.org/citation.cfm?id=1517494.1517517&coll=GUIDE&dl=GUIDE&type=series&idx=SERIES10716?=series&WantType=Proceedings&title=NetGames&CFID=71818071&CFTOKEN=71478144.

10. Michael Metzger, Birgit Zimmermann, Sonja Bergsträßer, Christoph Rensing, Ralf Steinmetz: Automating Layout Adaptation of Textual-based E-Learning Content. In: Proceedings of World Conference on Educational Multimedia, Hypermedia, June 2007.

11. Marek Meyer, Sonja Bergsträßer, Birgit Zimmermann, Christoph Rensing, Ralf Steinmetz: Modeling Modifications of Multimedia Learning Resources Using Ontology-Based Representations. In: Tat-Jen Cham and Jianfei Cai and Chitra Dorai and Deepu Rajan and Tat-Seng Chua and Liang-Tien Chia: Advances in Multimedia Modeling, vol. LNCS 4351, p. 34–43, Springer, January 2007. ISBN 978-3-540-69421-2.

12. Birgit Zimmermann, Sonja Bergsträßer, Christoph Rensing, Ralf Steinmetz: A Requirements Analysis of Adaptations of Re-Usable (E-Learning) Content. In: Piet Kommers and Griff Richards (Edits.): Proceedings of World Conference on Educational Multimedia, Hypermedia, p. 2096–2103, June 2006. ISBN 1880094592.

13. Christoph Rensing, Sonja Bergsträßer, Tomas Hildebrandt, Marek Meyer, Birgit Zimmermann, Andreas Faatz, Lasse Lehmann, Ralf Steinmetz: Re-Use and Re-Authoring of Learning Resources - Definitions and Examples. no. KOM-TR-2005-02, November 2005.

B.3 Betreute studentische Arbeiten

- Martin Ried: Untersuchung der Benutzerinteraktion und Möglichkeiten zur Unterstützung der Kooperation in MOGs
- Johannes Grimm: Entwicklung eines dynamischen Voice Communication Tools
- Vincent Krobisch: Einfluss und Wechselwirkungen von Communities auf MOGs
- Dirk Heeger: Modifikationen und Communities im Rahmen von MOGs
- Philip Henrizi: Übertragung von Konzepten aus Multiplayer Online-Spielen auf E-Learning
- Christian Gottron: Analyse und Bewertung der technischen Fragestellungen für ein Gaming Voice Communication Tool
- Jörg Daubert: Entwicklung einer Client-Applikation für die Erfassung und Übertragung von virtuellen Parametern
- Simon Poll: Konzeption einer Internet-Plattform für Spieler zur persönlichen Darstellung von Spielprofilen
- Christian Gottron: Qualitätsanalyse von Sprachkommunikation in mobilen Ad hoc Netzen

C Lebenslauf (Curriculum Vitae)

Persönliche Daten

Name:	Sonja Bergsträßer
Geburtsdatum:	27. April 1977
Geburtsort:	Heppenheim / Bergstraße
Nationalität:	Deutsch
Familienstand:	verheiratet

Akademischer Werdegang

2009 – heute	Technische Universität Darmstadt, Fachgebiet Multimedia Kommunikation (KOM) Wissenschaftliche Mitarbeiterin, Forschungsgruppe Knowledge Media
2006 – 2009	Technische Universität Darmstadt, DFG Stipendium Doktorandin im Graduiertenkolleg, Fachgebiet Multimedia Kommunikation (KOM)
2005 – 2006	Technische Universität Darmstadt, Fachgebiet Multimedia Kommunikation (KOM) Wissenschaftliche Mitarbeiterin, Forschungsgruppe Knowledge Media
2001	Nanyang Technological University, Singapore Studienarbeit, Centre for Advanced Media Technology (CamTech)
1997 – 2005	Technische Universität Darmstadt Studiengang: Informatik (Diplom) Abschluss: Diplom-Informatikerin

Schulische Ausbildung

1994 – 1997	Karl Kübel Schule, Gymnasium in Bensheim Abschluss: Abitur
1988 – 1994	Liebfrauenschule, Gymnasium in Bensheim
1984 – 1988	Hemsbergschule, Grundschule in Bensheim

Berufliche Tätigkeit

2006 – 2009	Hessisches Telemedia Technologie Kompetenz-Center (httc.e.V.) Wissenschaftliche Hilfskraft
2002 – 2003	Technische Universität Darmstadt, Fachgebiet Multimedia Kommunikation (KOM) Wissenschaftliche Hilfskraft, Forschungsgruppe Multimedia Semantics
2000 – 2002	Fraunhofer Institut für Graphische Datenverarbeitung Wissenschaftliche Hilfskraft, Abteilung Virtuelle Realität und Augmented Reality

D Erklärung laut §9 PromO

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, den 22. April 2010 Sonja Bergsträßer